Assessing text readability and quality with language models

Yang Liu

Helsinki February 12, 2020

Master's thesis

UNIVERSITY OF HELSINKI
Master's Programme in Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Studieprogram — Study Programme
Faculty of Science	Computer Science

Tekijä — Författare — Author
Yang Liu

Työn nimi — Arbetets titel — Title

Assessing text readability and quality with language models

Ohjaajat — Handledare — Supervisors
Dorota Glowacka and Alan Medlar

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master's thesis	February 12, 2020	57 pages + 0 appendices

Tiivistelmä — Referat — Abstract

Automatic readability assessment is considered as a challenging task in NLP due to its high degree of subjectivity. The majority prior work in assessing readability has focused on identifying the level of education necessary for comprehension without the consideration of text quality, i.e., how naturally the text flows from the perspective of a native speaker. Therefore, in this thesis, we aim to use language models, trained on well-written prose, to measure not only text readability in terms of comprehension but text quality.

In this thesis, we developed two word-level metrics based on the concordance of article text with predictions made using language models to assess text readability and quality. We evaluate both metrics on a set of corpora used for readability assessment or automated essay scoring (AES) by measuring the correlation between scores assigned by our metrics and human raters. According to the experimental results, our metrics are strongly correlated with text quality, which achieve 0.4-0.6 correlations on 7 out of 9 datasets. We demonstrate that GPT-2 surpasses other language models, including the bigram model, LSTM, and bidirectional LSTM, on the task of estimating text quality in a zero-shot setting, and GPT-2 perplexity-based measure is a reasonable indicator for text quality evaluation.

ACM Computing Classification System (CCS):
Computing methodologies → Artificial intelligence → Natural language processing
Applied computing → Document management and text processing

Avainsanat — Nyckelord — Keywords
Text readability, text quality, language models, LSTM, GPT-2, case study

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Contents

1 Introduction

Readability describes the complex relationship between a given text and the effort that readers make to understand it. Nowadays, people are increasingly relying on online materials to acquire knowledge. This greatly raises readers' expectations for the readability of materials. Take the largest Online Encyclopedia - Wikipedia as an example, although Wikipedia has been shown to be highly accurate in terms of factual content, it has been criticized for the quality of its writing, with many articles suffering from poor readability and style [1]. However, given the size of online texts, identifying article readability to enable not only the selection of appropriate materials for readers but editing suggestions for editors requires an automated approach.

A majority of the prior work in assessing readability has focused on identifying the level of education necessary for comprehension, which is generally expressed as the American grade level (i.e. years of primary and secondary education) [2, 3, 4, 5]. But the problem we want to explore here does not only assess readability in terms of comprehension, but in terms of style. Namely, do articles contain well-written, idiomatic prose that reads naturally to a native speaker of English. This is somehow similar to another task in literature called automated essay scoring (AES) that evaluates either contextual accuracy or writing quality of articles by automatically assigning goodness scores based on multiple criteria, such as word choices, narrativity, organization, and so on [6]. In this thesis, we studied these two aspects of readability, i.e., comprehension and style, which are referred to as readability assessment and text quality evaluation separately.

In this thesis, we propose two readability metrics: average per word perplexity and average likelihood ratio, based on the concordance of article text with predictions made using neural language models. Our goal is not to develop a classifier for "good" or "bad" writing, but to design metrics that allow us to rank documents, paragraphs and sentences by the degree to which text conforms to a corpus of documents that are assumed a priori to be highly readable and well-written. As both metrics are based on language models, they are unsupervised and do not require any special annotation.

In our experiments, the language model was trained using articles from The Guardian, a British newspaper, that we believe are well-written because they are written by professional journalists. Both metrics serve a dual purpose as they can be used to

(i) rank documents from the least to most readable, allowing us to identify which articles are most in need of attention in terms of writing quality and style and, (ii) use highlighting on a per word basis in the article to help an editor identify whether there are specific issues that should be fixed or that the entire article needs to be rewritten.

In this work, we make the following contributions:

1. We take an unsupervised ranking-based approach that identifies how well articles fit a model trained on well-written text. While existing approaches to assess the writing readability and quality are mostly based on supervised learning methods.

2. We demonstrate that our metrics are strongly correlated with text quality, i.e., whether an article is idiomatic and well-written, and have a weak correlation with text readability in terms of comprehension.

3. We explored the suitability of different types of language models, including a bigram model, LSTM, bidirectional LSTM, and GPT-2. We demonstrate that GPT-2 achieves the performance far beyond other language models on text quality assessment in a zero-shot setting.

This thesis consists of six chapters. Chapter 2 introduces the related works of both tasks i.e. readability assessment and text quality evaluation. In Chapter 3, we focus on the basis of language models and describes several widely used architectures in language modeling. Chapter 4 dives into the method we use for the task of text readability and quality evaluation. The description of datasets and language models used for performing experiments are also included here. In Chapter 5, we describe the experimental setup and analyze the results obtained. Chapter 6 contains the conclusion and suggestions for future work.

2 Related work

As we discuss the measurements of both text readability and quality in this thesis, it is necessary to introduce the related works and backgrounds in both fields. Therefore, section 2.1 will describe the related works of text readability assessment while we dedicated section 2.2 to the previous approaches to automated essay scoring.

2.1 Text readability assessment

Readability describes a complex relation between a given text and the efforts that readers make to understand it. The relation is considered to be associated with multiple factors, including lexical simplicity, presentation style, discourse cohesion, and background information [7]. Historically, interest in readability assessment in terms of comprehension originated in education and the military [8, 9]. Nowadays, in most cases, the research on readability assessment often identifies the level of education necessary for comprehension [2, 10]. Due to the high degree of cognitive complexity, readability assessment is considered as one of the most challenging tasks and therefore has attracted researchers' attention to explore automatic approaches for it [2, 3, 10, 11].

A variety of approaches were investigated to find readability measures that correlate well with human perception over the past few decades. Traditional readability measures were formulated by straightforward discrete features such as average sentence length, word length and difficulty and so on. Among them, Dale-Chall [12], Gunning's Fog (FOG, [8]), Flesch-Kincaid [9], and SMOG (Simple Measure of Gobbledygook, [13]) indexes, as the most representative traditional formulas, will be introduced in section 2.1.1. Instead of shallow features, some cognitively motivated features and discourse features were adopted to classify the readability level of texts, which will be described in section 2.1.2. As we mainly used language models to measure the readability in this thesis, we dedicated section 2.1.3 to the use of different language models on readability evaluation.

2.1.1 Traditional readability measures

Traditional readability formulas use shallow linguistic features, including the number of words, sentences and syllables, to output a readability score.

The first widely adopted readability measure, Dale-Chall readability formula (DCRF, [12]), was proposed in 1948. DCRF utilizes a manually curated list of easy to understand words, with all others deemed to be "difficult" words, to measure the text readability. It is calculated according to Equation 1:

$$DCRF = 0.1579(\frac{C(\textit{Difficult words})}{C(Total\ words)} \times 100) + 0.0496\frac{C(Total\ words)}{C(Total\ sentences)}, \quad (1)$$

where C is a counting function. If the percentage of difficult words is above 5%, we should add 3.6365 to the raw score to obtain the adjusted score. When assessing

texts, a higher Dale-Chall score indicates that the text is more difficult to read.

Subsequently, in 1952, Gunning [8] proposed the Gunning fog index (GFI), in order to estimate the grade level required to understand text on the first reading. It was formulated as follows:

$$GFI = 0.4 \times [\frac{C(Total\ words)}{C(Total\ sentences)} + 100\frac{C(Complex\ words)}{C(Total\ words)}], \tag{2}$$

where *Complex words* refers to words that consists of three or more syllables. Intuitively, the higher the GFI index, the less readable the text.

In 1975, Flesch-Kincaid readability tests were used to assess the readability of technical manuals in the United States military [9]. There were two tests: the Flesch Reading Ease Score (FRES) and the Flesch-Kincaid Grade Level (FKGL). Both tests utilized the total number of syllables, words and sentences to approximate the reading ease score and corresponding grade level of a text. However, they employed different weighting factors obtained by performing regression on a dataset of Naval school passages [9]. Their formulas are listed below,

$$\begin{aligned} FRES &= 206.835 - 1.015\frac{C(Total\ words)}{C(Total\ sentences)} - 84.6\frac{C(Total\ syllables)}{C(Total\ words)}, \\ FKGL &= 0.39\frac{C(Total\ words)}{C(Total\ sentences)} + 11.8\frac{C(Total\ syllables)}{C(Total\ words)} - 15.59. \end{aligned} \tag{3}$$

In the Flesch Reading Ease test, the higher FRES implies that the text is easier to read. While in the Flesch–Kincaid Grade Level test, FKGL directly corresponds to a U.S. grade level.

Similar to FKGL formula, SMOG (Simple Measure of Gobbledygook, [13]), is used to estimate the years of education needed to comprehend the given texts adopting the number of polysyllables and sentences as features. SMOG is calculated according to the following formula:

$$SMOG = 1.0430\sqrt{C(Polysyllables) \times \frac{30}{C(Total\ sentences)}} + 3.1291, \tag{4}$$

where $C(Polysyllables)$ means the number of words that contain more than three syllables. In practice, SMOG is only applicable if the text has more than 30 sentences as it was normed on 30-sentence samples.

All readability assessment formulas cited above operate at the level of document, whether that is a technical manual, book or web page. Based on the above formulas, the use of shallow features will return the same score if we shuffled the order of

words in each sentence. This suggests that traditional readability measures are easily cheated. In addition, the parameters of these formulas were obtained using some specific dataset and might not be transferable to other datasets [14].

2.1.2 Cognitive and discourse features

To overcome the deficiencies of traditional readability measures, multiple approaches that used complex syntactic and semantic features to predict the readability level have been proposed. These approaches usually require extensive feature engineering and, therefore, can achieve much better results in readability assessment.

Feng et al.(2009, [3]) produced an automatic readability metric that is tailored to the literacy skills of adults with intellectual disabilities (ID). They investigated multiple discourse-level, cognitively motivated features that correlate with the readability level for adults with ID, such as total number of nouns and name entities, number of unique entities, and the average lexical chain span. They argued that the use of audience-specific features can improve the accuracy of readability assessment on the specific user group.

Recently, Mesgar and Strube (2018, [11]) proposed a local coherence model that identifies whether adjacent sentences are semantically similar. They adopt a recurrent neural network (RNN) to represent semantic information that connects two adjacent sentences and then employed a convolutional neural network (CNN) to encode the pattern of semantic information changes across sentences. As a result, the model achieves new state-of-the-art results on the task of readability assessment.

Although the use of cognitive and discourse features improves the accuracy of readability assessment, it has been criticized for the requirement of sophisticated manual feature engineering [15].

2.1.3 Language models

It is well noted that language models can capture linguistic features well and, therefore, have been widely used to evaluate text readability.

Initially, researchers in computational linguistics have adopted statistical language models that extract context information from texts to improve the accuracy of readability assessment. In 2001, Si and Callan [2] classified scientific web pages of different grade levels based on a unigram language model. They built a reading level

classifier by linearly combined a unigram model and surface linguistic features. It was shown that their method surpasses the widely used Flesch-Kincaid readability formula.

In the work of Schwarm and Ostendorf (2005 and 2009, [10, 16]), SVM-based detectors that incorporate the perplexity from unigram, bigram and trigram language models and other traditional features were created to perform readability assessment. They investigated the influence of syntactic features and traditional lexical features on estimating accuracy and explored the alternative regression model to the SVM classifier in the context of limited annotated training data.

Later on, the introduction of neural language models has achieved better results for the task of readability assessment. Azpiazu and Pera (2019, [15]) proposed a multi-attentive long-short term memory (LSTM) architecture called Vec2Read for automatic multilingual readability assessment. The model leveraged a multi-attentive strategy to make the network concentrate on specific words, sentences, and paragraphs that correlated with reading levels and then predicts probability over reading levels based on attention scores. The use of attention mechanism yields better results than the same LSTM architecture with traditional and deep learning strategies.

Martinc et al. (2019, [7]) presented both supervised and unsupervised approaches with multiple novel neural language models, such as LSTM and BERT (Bidirectional Encoder Representations from Transformers, [17]), on readability estimation. It was shown that the supervised approach they employed achieved better results than traditional readability formulas and is transferable across languages. It is worth mentioning that they directly used model perplexity as a metric for assessing readability on an unsupervised setting, which coincided with our idea. However, their perplexity-based measures were concluded as the worst measurements according to their experimental results. In this thesis, we argue that perplexity is a strong indicator of text quality evaluation.

2.2 Text quality assessment

Text quality assessment is a task to evaluate how well-written an essay or article is by assigning a goodness score to it, which is often referred to as AES. AES aimed to relieve the heavy work loads of educators or teachers in assessing large amounts of essays and can be traced back to the 1960s when the first AES system Project Essay Grade (PEG) was proposed [18]. Since then, researchers have considered AES as a

regression, classification and preference ranking problem and utilized uncountable statistics, machine learning and deep learning techniques to approach it [19, 20].

We can classify existing approaches for AES according to the level of NLP (short for Natural language processing) techniques applied [6]. Based on this, three categories could be identified: **shallow NLP** refers to statistical techniques for shallow linguistic features, i.e. lexical level features, **full NLP** involves more sophisticated NLP techniques for not only lexical level but syntax, semantics and structure level features, **deep learning NLP** involves advanced deep learning architectures, such as RNN, CNN and neural attention models for automatic feature selection [6, 20, 21].

Therefore, in this section, we will follow these three categories to briefly introduce the development of AES systems and approaches.

2.2.1 Shallow NLP techniques

The early AES systems mainly used statistical analysis of one or multiple shallow linguistic features, i.e. lexical level features, such as keyword frequency or punctuation counts. The PGE proposed by Page et al.(1966, [18]), the Intelligent Essay Assessor (IEA) introduced by Thomas et al.(1997, [22]) and E-rater presented by Burstein et al.(1998, [23]) are three of the most representative AES systems in this category.

PEG predicted essay scores using regression with up to 30 parameters, such as average word length, number of semicolons, and word rarity [18, 24]. As the first AES system, despite critics arguing that computers cannot determine the writing competence as well as humans does, it proved that the PEG score correlated well with scores by human raters. However, PEG can only capture writing style rather than contents of essays as only shallow linguistic features were used. Despite this, PEG has been widely used due to its conceptual simplicity and limited computer requirements[21, 25, 26].

IEA used latent semantic analysis (LSA, [22]) to grade essays automatically, which was first designed for information retrieval to measure the similarity between documents. The underlying principle of LSA is to identify the semantic similarity between the input document and indexing documents by converting documents into vectors based on the frequency of indexed terms in the input document. Then, the IEA system will assign a goodness score using the average score of several most similar calibration documents. The number of calibration documents used is predefined as

a hyper-parameter. Contrary to PEG, IEA can capture contents of essays instead of writing style due to the nature of the LSA method. It is also reported a high correlation between IEA scored and human scored essays [26].

Compared to PEG and IEA, E-rater has more sophisticated architecture which contains three modules corresponding to syntactic variety, discourse structure, and content analysis. Syntactic variety was represented using the ratio of complement, subordinate, infinitive, and relative clause and occurrences of modal verbs per sentence per essay. Discourse structure was similar to PEG but with up to 60 variables. Content analysis aimed to measure the semantic similarity between essays, which shares the same concept as the IEA system. It was shown that E-rater correlates significantly with human raters [23]. E-rater is the superior choice for grading content and has been used to score the General Management Aptitude Test (GMAT) [23, 26].

Other existing approaches, such as Larkey's system (1998, [27]), formulated AES as a classification task and applied the classical Naive Bayes model. Further, multinomial Bernoulli Naive Bayes was used to classify the categories of text quality[28]. Despite the fact that shallow features can hardly capture the semantic information, it is consistently reported that multiple systems or approaches using shallow NLP techniques could achieve 60% exact agreements with human raters [21, 6, 26].

2.2.2 Full NLP techniques

Due to the limitation of shallow features, early systems could not capture the discourse cohesion and semantic information when evaluating the text quality. As a result, they are more likely to be tricked by some nonsense essays generated intentionally. For example, IEA could be fooled to give a high score to a text that consists of a sequence of relevant words without any structure [26].

Therefore, to eliminate the limitation, more complicated NLP tools were applied into AES systems and approaches, including syntactic analyzers, rhetorical parsers, and semantic analyzers. Rhetorical parsers aimed to determine the discourse structure of texts [29]. Syntactic analyzers were used to identify sentence components and analyze syntactic dependencies between them [30]. While semantic analyzers were utilized to identify the function of each component in the text [6, 31].

C-rater (2001, [23]) and its advanced version E-rater (2006, [32]) are two most popular systems that were benefited from these NLP tools. C-rater and E-rater

are automated scoring systems that were designed for Educational Testing Service (ETS), the largest nonprofit educational testing and assessment organization in English. Both of them made use of not only the lexical level features but more complex syntactic features extracted from the training texts. As a result, they achieved high percentage agreements (over 80% agreement) with human raters in both small-scale and large-scale studies [6].

Subsequently, a new series of ranking-based methods were proposed, combined with complex NLP tools. In 2011, Yannakoudakis et al. formulated the AES task as a preference ranking problem [33]. It made use of hybrid features containing word n-grams, part-of-speech n-grams, grammatical complexity, and parsing trees to rank the order of essays based on the writing quality and then measured the correlation between system scores and human graders' scores. Further, Chen et al. [34] applied a list-wise ranking technique into their system in order to reduce biases between human raters and computer based systems by directly incorporated the agreements term into the loss function. The features they utilized included syntactic variety, sentence fluency, and prompt-specific features.

2.2.3 Deep learning techniques

Despite the fact that full NLP techniques based systems achieved fantastic performance on AES, they required sophisticated manual feature engineering [6]. To combat this issue, people began to explore more advanced approaches that made use of state-of-the-art deep learning techniques to learn syntactic and semantic features automatically.

It should be mentioned that there was no existing benchmarks that could be used to uniformly evaluate AES systems until Shermis and Burstein presented the Automated Student Assessment Prize (ASAP) dataset in 2013 [19]. Subsequently, with the emergence of powerful deep learning tools and the benchmark dataset, significant progress have been made on the AES task.

In 2016, Alikaniotis et al. adopted a LSTM based language model to perform AES. The model took the word embeddings that were learned by the word contributions to text score as input, and obtained the essay expression using the last hidden states of a two-layer bidirectional LSTM. The LSTM based model achieved excellent results, outperforming other commonly used systems [35]. Further, Taghipour and Ng [36] explored several neural network models including LSTM, LSTM with attention

mechanism, CNN combined with LSTM and bidirectional LSTM for the AES task. Their method avoided a large amount of effort to perform feature engineering and the best architecture, LSTM, outperformed a strong baseline by 5.6% in terms of the Kappa coefficient.

Subsequently, CNN were also adopt for AES. Dong and Zhang (2016, [37]) employed a hierarchical CNN on the AES task. It split texts into sentences and extracted both sentence-level and text-level information using a two-layer CNN to obtain the text representation. This method was demonstrated to outperform the baseline approaches, such as Bayesian Linear Ridge Regression (BLRR) and Support Vector Regression (SVR), showing automatically learned features to be competitive to handcrafted features. Dong et al.(2017, [20]) systematically investigated CNN and LSTM on sentence-level and text-level representation respectively, and the effectiveness of attention mechanism on automatically selecting more relevant n-grams and sentences for the task. It was proved that the method outperforms the previous methods and attention mechanism is particularly effective in the AES tasks.

Our work combines neural networks with an unsupervised ranking-based method to provide further help for writing and editing. Results shows that our metrics strongly correlate with quality scores graded by human raters.

3 Language models

Language modeling is an approach to understanding linguistic structures within the language by learning from corpus data. The studies of language structures can be traced back to two centuries ago [38]. Since then, linguists have accumulated a large number of corpora to find syntactic rules of the language. These rules were formalized into the grammar that we know today [39]. While grammar can indeed recapitulate the structure and usage of the language, they cannot completely identify all situations that occur in language use as people may sometimes speak ungrammatically to meet their daily communication needs [40]. Therefore, in order to address this problem, scientists have instead made use of statistics as a tool to identify the common patterns within languages. More specifically, language models could learn the probability distribution of words that occurred in a sequence from corpus data. This kind of approach is called language modeling [41].

Language models are probabilistic models that assign probabilities to words or sequences. More specifically, given a corpus C, language models can learn the condi-

tional probability distribution $p(w|c)$, the probability of a word w appearing in the context c. According to this, language models can be used to predict the probability with each possible next word occurred and generate text by continuously sampling from the probability distribution [38].

There are two main categories of language models: statistical language models (SLM) and neural network based language models (NNLM). SLM was first proposed in 1980, and then widely used in a variety of NLP tasks that rely on probability of words or sequences [42]. For example, statistical language models were used to assign a goodness score based on the probability that a sequence occurred and therefore could suggest the most accurate translation for machine translation tasks or the most likely transcription in speech recognition [43, 44, 45, 46]. However nowadays, powerful neural networks based language models have replaced the statistical models and become dominant in NLP. These neural language models have achieved state-of-the-art performance on a large amount of challenging NLP tasks like machine translation, speech recognition, and natural language inference [47, 48, 49, 50]. Here, we aim to introduce our new word-level metrics to assess the text readability and quality. As the metrics we proposed are actually mathematical variants of the word probability that language models output, it is greatly necessary to introduce the theoretical background of language models before we delve into the details of the metrics. In this chapter, the details of language model basics and the formulas as well as architectures of multiple language models will be described.

3.1 Language model basics

In this section, we will introduce the flow of building a language model and its evaluation, in order to help readers understand model details better. Section 2.1 describes language representation and the training process, while section 2.2 depicts evaluation methods of language models in detail.

3.1.1 Introduction

Building a language model often requires three compulsory steps, which are processing the corpus, training the language model, as well as fine-tuning and evaluating the trained model. Assuming that we have a clean corpus that is ready for training and we want to build a language model based on it, the first thing to do will be splitting the text into tokens and build a **vocabulary** which is a list of all unique

tokens in the corpus. This step is called **tokenization**.

Then, before feeding data into a language model, it is essential to encode tokens into a readable format for a language model. However, different models require different inputs. For statistical language models, encoding is not necessary. As for neural language models which require the input sequence to be a vector or matrix, there are two commonly used ways to encode tokens, which are called **one-hot encoding** and **word embedding**.

One-hot encoding is a method to convert each token into a unique one-hot vector. One-hot vector consists of all 0s, except for a single 1 in some position. For a vocabulary of size N, the n-th one-hot vector is the vector whose n-th entry is 1 and all other entries are 0s. Therefore, there would be a total of N one-hot vectors that could represent all N unique tokens. These one-hot vectors could be either directly fed into models or stacked as batches. One-hot encoding is easy to make and highly interpretable for language model use. However, it leads to a higher level of sparsity as the vocabulary size increases [41].

Another popular approach for encoding is word embedding, which was first introduced in the 1960s [51]. Word embedding represents a set of techniques where distributed representations of individual tokens are learned based on the usage of tokens [52]. It is used to map unique tokens into vectors with a fixed length. These vectors obtained by statistical or machine learning methods are supposed to capture the semantic similarities between linguistic items. Word embedding does not only address the sparsity issue of one-hot encoding but also provides semantic information for language models. Therefore, a large amount of language models and downstream NLP tasks take word embedding as the very first step to conduct [52, 53].

3.1.2 Evaluation of language models

Intuitively, we can evaluate the performance of a language model by measuring the improvements that the model made on a specific NLP task. For example, in speech recognition, we can compare the output of the speech recognizer with different language models to see which language model leads to a more concise transcription. This kind of end-to-end evaluation is called **extrinsic evaluation** [41]. Extrinsic evaluation is considered to be the best way to evaluate language models because they can better serve the tasks we are performing [41].

However, testing language models on larger downstream NLP tasks is often very

expensive as those downstream NLP applications often have more complex architectures [41]. Therefore, it is essential to instead create **intrinsic evaluation** metrics that are independent of any application and directly measure the performance of a language model. The most commonly used metric is called **perplexity**.

It is commonly used to evaluate a language model by computing the average perplexity on the test set. The perplexity describes the degree of uncertainty that the model feels about the sequence of interest. It is defined as the normalized inverse of the probability of that sequence under the language model. For a sequence of words $W = w_1 w_2 ... w_N$ that of size N, the perplexity (PP for short) is formulated as [41]:

$$PP(W) = \sqrt[N]{\frac{1}{p(w_1 w_2 ... w_N)}}. \tag{5}$$

According to Equation 5, we can conclude that the higher the probability of a sequence, the lower the perplexity. Basically, we expect that the language model is confident of sequences that appeared in the corpus on which the model was trained. That is to say, the probabilities of sequences in the corpus should be high and the perplexity of those sequences should be low.

3.2 Statistical language models

Statistical language models directly model the conditional probability distribution of a word given the words that precede it. That is to say, given a sequence of words $w_1 w_2 ... w_m$, SLMs will compute the probability of $p(w_n | w_1 w_2 ... w_{m-1})$ by calculating the ratio of the number of times that the history sequence $s_{old} = w_1 w_2 ... w_{m-1}$ followed by word w_m to the number of times that s_{old} occurs in the given corpus, as shown in Equation 6 [41].

$$p(w_m | w_1 w_2 ... w_{m-1}) = \frac{C(w_1 w_2 ... w_{m-1} w_m)}{C(w_1 w_2 ... w_{m-1})}, \tag{6}$$

where the capital C stands for a counting function.

SLMs can also be used to compute the joint probability of an entire sequence according to the **chain rule of probability** [41]:

$$p(w_1 w_2 ... w_m) = p(w_1) p(w_2 | w_1) p(w3 | w1w2) ... p(w_m | w_1 w_2 ... w_{m-1}). \tag{7}$$

With the ability to assign probabilities to words or sentences, SLMs are widely used in speech or optical character recognition, spelling or grammatical error correction as well as machine translation tasks [39, 41].

It is intuitive that the relation between the very beginning word w_1 and the last word w_m will be extremely weak as the length of the sequence increases. Therefore, we assume that, for a word w_m, its probability of appearing in the end of the sequence $w_1w_2...w_{m-1}$ only depends on n previous words rather than all previous words. Based on it, Equation 6 and 7 will be converted into:

$$p(w_m|w_1w_2...w_{m-1}) \approx p(w_m|w_{m-n}...w_{m-1}), \tag{8}$$

$$p(w_1w_2...w_m) = p(w_{m-n})p(w_{m-n+1}|w_{m-n})...p(w_m|w_1w_2...w_{m-n}). \tag{9}$$

This kind of models are considered to have a **sliding window** of size n and are called **n-gram language models**. Here, the **n-gram** is responsible for a sequence of size n. Similarly, the **bigram** (or 2-gram) represents a two-word sequence and **trigram** (also called 3-gram) stands for a sequence of three words or tokens [41].

As the size of the sliding window n increases, the complexity of the model and the model size increases as well. For a corpus that has N unique tokens, if we want to model the conditional probability that a sequence of size $n-1$ followed by a word w_n, we have to count how many times that N^n n-grams and N^{n-1} (n-1)-grams appear in the corpus according to Equation 8 and 9. As a result, the model size will be extremely large when n is a large number. Therefore, n is often smaller than 5 in practice.

When $n = 1$ (the uni-gram model), it is suggested that the probability of each word only depends on its own probability in the given corpus. The Equation 9 is simplified as $p(w_1w_2...w_m) = p(w_1)p(w_2)...p(w_m)$. Despite the fact that the uni-gram model is easy to use, it cannot capture contextual information compared to higher dimension language models. The bi-gram model computes the conditional probability of a sequence $w_1w_2...w_{m-1}$ followed by a word w_m by the conditional probability of a word w_n appearing after its previous word w_{n-1}. Note that the bi-gram model is a variant of the **Markov model**, which is a kind of classical probabilistic model used for modeling randomly changing systems [41].

Despite the fact that model complexity increases as window size n and vocabulary size N increase, it is understandable that higher-order n-gram models will do a better job on language modeling on corpora that have suitable sizes. However, other than the large increase of model complexity, higher-order n-gram models have another problem called the **curse of dimensionality** [41, 39]. This term describes the phenomenon that sparsity of the corpus increases as the model dimensionality increases. Specially, it is difficult to accurately estimate the probability of n-grams

that rarely or never appeared in the corpus, because those n-grams may be just missing from the corpus as any corpus is limited. Therefore, it is not reasonable to assign them an extremely low or even zero probabilities with the consideration of generalization.

The method used to solve the sparsity problem in SLMs is called **smoothing**, which gets rid of the extremely low or zero probabilities by adjusting the maximum likelihood estimator of language models [41]. Smoothing is utilized to deal with the problem that n-grams did not appear in the training corpus but are in the unseen test dataset. Before that, how do we address words from the test corpus that are not in the vocabulary? Normally, we replace those words with an **unkown** marker such as <UNK> and then treat <UNK> as a normal word. After that, we use smoothing to solve those n-grams that the model did not see in the training corpus.

The most used smoothing methods are **Laplace smoothing**, **add-k smoothing**, **stupid backoff** and **Kneser-Ney smoothing**. Laplace smoothing (also called add-1 smoothing) is to add 1 to all counts of n-grams. Using bi-gram model as an example, the conditional probability is modified as follows, where N represents for the number of bi-grams in the corpus [41].

$$p(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}, \tag{10}$$

$$p(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+N}. \tag{11}$$

In this way, all zero probabilities are convert to a small number $\frac{1}{N+V}$, assuming that the vocabulary has size V. This is of benefit for estimating the probabilities of sequences in test dataset. However, it is suggested that Laplace smoothing is not a good choice for language models, because it will assign the same probabilities to each unseen token. Despite this, it is considered as a baseline for other smoothing methods in SLMs [41].

Add-k smoothing shares the same concept with Laplace smoothing. It instead adds k to counts for n-grams as shown in the following [41]:

$$p(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+k}{C(w_{n-1})+k*N}. \tag{12}$$

It is often beneficial to fine-tune k by optimizing it on a validation set. Therefore, add-k smoothing is considered a more flexible approach than add-1 smoothing. However, add-k smoothing suffers from the same deficiency of generating probabilities with poor variance as the add-1 smoothing does [41, 54].

The stupid backoff smoothing is quite a different approach compared to add-1 and add-k smoothing, which substitute the probability of an n-gram with the probability of its corresponding (n-1)-gram if the n-gram rarely appears in the corpus. Taking trigram model as an example, assuming that we want to compute the conditional probability $p(w_n|w_{n-1}w_{n-2})$ but the sequence $w_{n-2}w_{n-1}w_n$ are missing from the corpus. What stupid backoff does is estimate the probability using the bigram probability $p(w_n|wn-1)$. This step is called backoff. Variants of stupid backoff, such as **Katz backoff** and **Good-Truing backoff** generally perform well in practice but fail under some conditions. For example, if the sequence $w_{n-2}w_{n-1}w_n$ rarely appears in a corpus, it is possible that the sequence is grammatically incorrect. However, the backoff-based approaches may consider that the corpus is too limited and therefore cause an unsatisfactory result [39, 41].

The most widely used method for smoothing is called Kneser-Ney smoothing. The basic idea behind this method is to discount the probability mass of n-grams by subtracting some discounted rates d_m, where d_m is the difference of n-grams counts in training and test sets. That is to say, if all n-grams that occur m times in training set and appear m' times on average in the test set, d_m will be computed by $m - m'$. Other than the discounting issue, the Kneser-Ney method also solves the problem that some unigrams have high probability only because their corresponding n-grams have large counts. It assumes that the continuation probability $p_{con}(w_i)$ of a word w_i is proportional to the number of n-grams that w_i has appeared in. Final formulas of Kneser-Ney smoothing for the bigram model are as follows [41]:

$$p_{kn}(w_i|w_{i-1}) = \frac{max(C(w_{i-1}w_i) - d, 0)}{C(w_{i-1})} + \lambda(w_{i-1})p_{con}(w_i) \tag{13}$$

$$p_{con}(w_i) = \frac{|v : C(vw_i) > 0|}{\Sigma_{w'_i}|v : c(vw'_i) > 0|}, \tag{14}$$

where v represents some token in vocabulary V, while λ stands for the normalizing constant of the continuation unigram model.

3.3 Neural network based language models

Despite the fact that SLMs such as n-gram model is highly interpretable and easy to use, they suffered from the issue of sparsity and model complexity. In spite of applying smoothing techniques and constraining the model degree, it is still difficult for SLMs to capture the language patterns of given corpora due to the limitation that only simple models can be used and the linear nature of the model itself.

With the development of deep learning, researchers have started to consider applying **neural networks**, the core of deep learning, into language modeling. In 2001, the first neural network language model was proposed by Bengio et al. [51], which obtained the performance far beyond the SLMs and addressed the curse of dimensionality head on. From then, more and more deep learning based architectures were invented to perform language modeling, such as CNN [55], RNN [56], and LSTM [57]. More impressively, bidirectional language models with the concept of collecting information from both left-side and right-side context were introduced in order to capture the language pattern better [58]. Recently, large pre-trained language models such as GPT-2 [59] and BERT [17] have been shown to surpass the previous language models and also perform well on various challenging downstream NLP tasks, such as natural language inference, question answering, and machine translation.

In this section, we will give a brief introduction to neural networks, basic NNLMs, such as RNN and LSTM based language models, and pre-trained language models, including GPT-2 and BERT. As the prerequisite of pre-trained models, the attention based architecture – **Transformer** will also be described here.

3.3.1 Neural networks

A neural network is a network of computing units called neurons. The name 'neuron' was inspired by biology due to the similar functionality of the computing units and human neurons as computational elements. Each computing unit takes vectors as input, performs a transformation on it, and produces single output values. The output y is calculated according to the following formula [60]:

$$z = \mathbf{w} \cdot \mathbf{x} + b \tag{15}$$

$$y = \sigma(z), \tag{16}$$

where $\mathbf{x}$ is the input vector and $\mathbf{w}$ stands for the weight vector, while b is the bias term. The Equation 15 performs the linear transformation that maps the input vector x to a scalar z. The transformed scalar z is obtained by summing up the dot product $\mathbf{x} \cdot \mathbf{w}$ and the bias term b. The $\sigma(.)$ in Equation 16 represents a non-linear **activation function**, which defines the output value of the neuron.

The selection of activation function plays an essential role when implementing neural networks and strongly depends on the task of interest. There are three popular activation functions, including the **sigmoid**, the **tanh**, and the **ReLU** (rectified

linear unit). The sigmoid function outputs values ranging from 0 to 1 and is mainly used for classification tasks. The tanh maps the output into the range of $[-1, +1]$, and is considered better than the sigmoid function in practice as its output is zero centered which makes optimization easier. The most straightforward activation function, ReLU, outputs x when x is positive, and 0 otherwise. Three functions are presented in Equation 17, 18 and 19 respectively. In modern neural networks, the default recommendation is to use the ReLU [41].

$$sigmoid : y = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{17}$$

$$tanh : y = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{18}$$

$$ReLU : y = \max(x, 0) \tag{19}$$

The quintessential example of neural networks is the **feedforward network** [60]. The feedforward network is a multi-layer neural network that performs computation iteratively from one layer to the next without passing back. The structure of a three-layer feedforward network is shown in Figure 1 [61]. The network is fully-connected, which means that neurons between adjacent layers are pairwise connected [61].

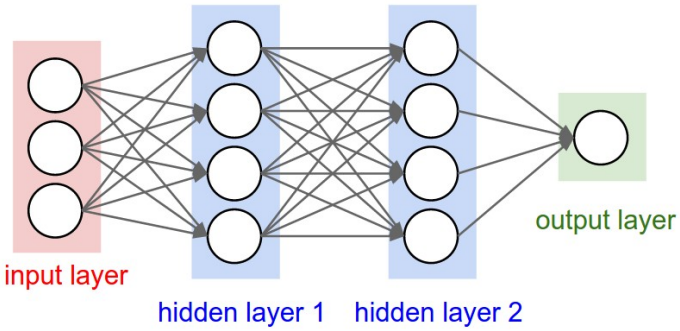

Figure 1: A simple three-layer feedforward neural network: the network contains one input layer, two hidden layers and one output layer. Each layer consists of multiple units, represented by circles [61].

The core of the neural network is the hidden layer which takes a weighted sum of its input and then apply a non-linear transformation to it [41]. The computation of feedforward networks proceeded according Equation 20. Here, $\mathbf{x}$ and $\mathbf{y}$ stands for the input and output vectors of the network separately, the superscript l represents the l-th layer, $\mathbf{W}^{(l)}$ and $b^{(l)}$ refer to the weight matrix and bias term of the l-th layer, respectively, $\sigma(\cdot)$ means the activation functions, and the capital L is the number of layers in the network. Note that each hidden layer could apply distinctive activation

functions.

$$\begin{aligned} \mathbf{z}^{(0)} &= \mathbf{x} \\ \mathbf{z}^{(l)} &= \mathbf{W}^{(l)} \cdot \mathbf{z}^{(l-1)} + b^{(l)} \\ \mathbf{a}^{(l)} &= \sigma(\mathbf{z}^{(l)}) \\ \mathbf{y} &= \mathbf{a}^{(L)} \end{aligned} \tag{20}$$

The network shown in Figure 1 has a single output node and can perform binary classification if the activation function of the last layer is the sigmoid function. However, when trying to do a multi-class classification task, the **softmax** function would be a priority choice. The softmax function can output normalized vectors of real values that encode a probability distribution. Based on it, it is straightforward to assign the most likely category to an input vector. The softmax function is presented below:

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{d} e^{z_j}} \quad 1 \leq i \leq d \ , \tag{21}$$

where d is the number of categories, z_i and z_j represents for i-th and j-th entry of vector z.

After knowing the structure of the feedforward neural network, the next step is to learn how to train the network. Training neural networks means to find wight matrices and bias terms (parameters of neural networks) that fits the training data most appropriately. From this aspect, neural networks were considered as supervised learning approaches. Therefore, we use the same method that is used in other supervised learning models to find the most likely parameters. The first step is to define a loss function, and then, we minimize the loss function using the **gradient descent** optimization algorithm.

However, performing gradient descent requires the knowledge of gradient of the loss function whose analytical expression is hard to get. The solution to this is an algorithm called **error backpropagation**. The method computes the derivatives of intermediate variables, such as $\mathbf{z}^{(l)}$ and $\mathbf{a}^{(l)}$ with respect to model parameters and then makes use of chain rule to obtain the derivative of the model loss L with respect to model parameters.

Finishing the model training and evaluation, we can utilize the trained neural networks to make predictions. Before that, there are some practical techniques that we should pay attention to. Firstly, it is not allowed to initialize the gradient descent with all the weights and biases assigned the value 0, because doing this will make

hidden units symmetric and the model loses its non-linearity [60]. As a result, we have to initialize the parameters with small random numbers. Secondly, it is always helpful to fine-tune the hyper-parameters which are determined by the architecture designer, such as the number of hidden units and layers [41]. In addition, sometimes the architecture of neural networks we chose might be too complex to fit the data, which leads to the problem of **overfitting**. One technique that is popularly adopted to avoid overfitting is called **dropout**, which randomly drops some units and their connections from the network during training [41].

3.3.2 Neural network language model

The concept of applying neural networks in language modeling was first proposed in 2001 by Begino et al [51]. The goal of their work is to fight the curse of dimensionality of SLMs by learning a distributed representation for tokens. At the same time, the model learns a probability distribution of word sequences in terms of the token representations. The architecture of the neural language model is shown in Figure 2 [51].

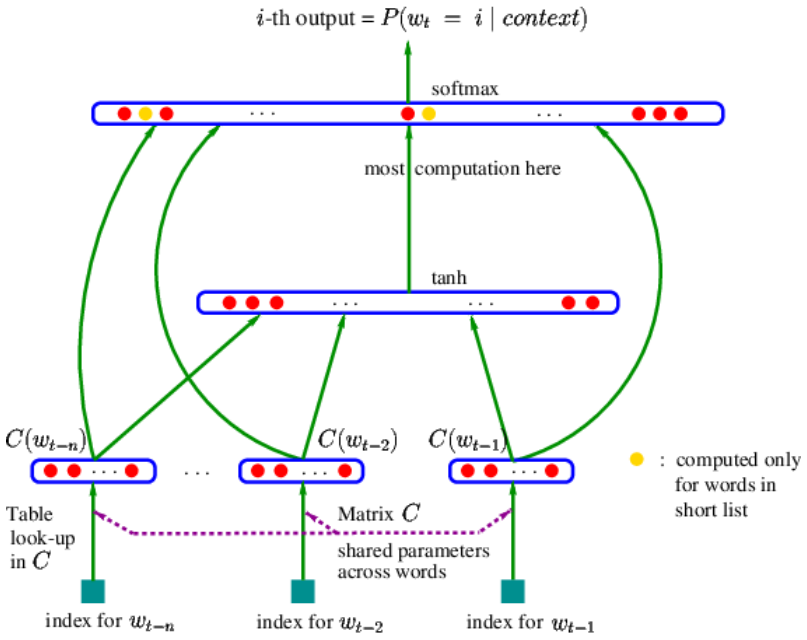

Figure 2: The architecture of a probabilistic neural language model: the neural language model contains a embedding layer that is represented as Martix C, a hidden layer with tanh activation function, and an softmax output layer [51].

According to Figure 2, The embedding layer transforms the index of each input

word into a feature vector of a predefined dimension. The resulting word vectors are concatenated as the input of the tanh hidden layer. Thereafter, the softmax layer maps the output vector of the hidden layer into a conditional probability distribution over words in the vocabulary V. If we set an input sequence $w_1 w_2..w_T$, a mapping function C that assigns feature vectors to words in V, and a function $\mathbf{g}$ that outputs the conditional probability distribution $p(w_t|w_1..w_{t-1})$, the model can be formulated mathematically in Equation 22.

$$p(w_t|w_1..w_{t-1}) \approx p(w_t|w_{t-n+1}..w_{t-1}) = g(C(w_{t-1}), ..., C(w_{t-n+1})) \tag{22}$$

It was shown that neural language models surpass n-gram language models by a high margin [51, 56]. However, the model suffers from two major deficiencies: 1) it fails to deal with the sequence of varying lengths and, 2) it is impossible to capture important temporal properties of the language. To solve these problems, the recurrent neural networks (RNN) were adopt by researchers in computational linguistics.

3.3.3 Recurrent neural network based language model

Unlike feedforward neural networks, the recurrent neural architecture can recursively process and leverage the information from the past when computing the current state, and therefore have been widely used for modeling sequential data. Particularly, RNNs have become a prominent choice for language modeling.

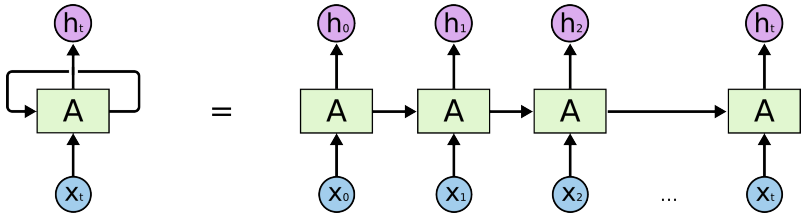

Figure 3: The structure of a general recurrent neural network: the left diagram depicts the structure of a general RNN, which can be unfolded into a chain diagram on the right side. An RNN contains a chunk of neural networks, A, which takes x_t and the past hidden state h_{t-1} as input and outputs the current state h_t [62].

Figure 3 [62] illustrates the structure of a general RNN. The loop represents the flow of information passing from one time step to the next. Mathematically, the computation of the RNN is based on Equation 23 [60, 63].

$$h_t = f(\boldsymbol{W} h_{t-1} + \boldsymbol{U} x_t + b) \ , \tag{23}$$

where $f(.)$ represents the activation function of the neural network A, while matrices $\boldsymbol{W}$ and $\boldsymbol{U}$ stand for the weight matrices for both input x_t and previous hidden state h_{t-1} in the hidden layer. It should be noted that a softmax layer that maps the output h_t to a conditional probability distribution will be added when applying RNN into language modeling.

The intrinsic property of recurrence allows RNN to not only capture the temporal nature of language but to eliminate the limitation on the length of input sequences. This leads to incredible success in NLP [60]. However, despite the fact that RNN is supposed to learn the long-term dependencies between the previous and recent information, it was shown that the standard RNN is not capable of doing that as the sequence length increases due to the **vanishing gradient** problem [63]. As a result, the long short term memory (LSTM) network was proposed to overcome the issue.

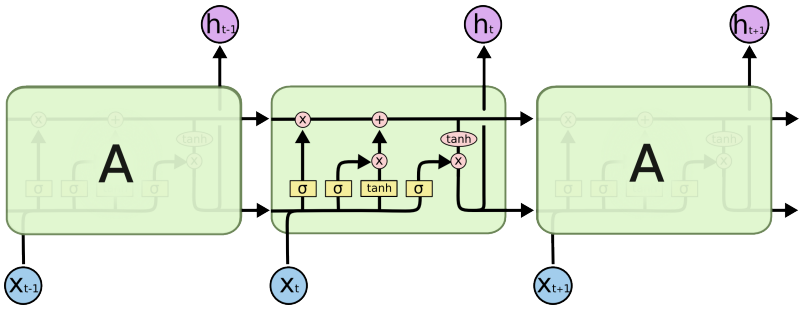

Figure 4: The structure of a LSTM network: the symbol of σ within a yellow box stands for the operation of passing a gate. These symbols represent the input gate, the forgetting gate, and the output gate from left to right [62].

LSTM shares the main concept from plain RNNs. As one of the most used variants of RNN, LSTM considered more about the way to process the memory of previous hidden states by adding three control gates: the input gate, the forgetting gate, and the output gate respectively. The diagram of LSTM is shown in Fig.4 [62].

Combined with a set of formulas of LSTM (Equation 24), we illustrate the process flow of an LSTM network here. Firstly, the states of three control gates are determined by the linear combination of input x_t and the previous output h_{t-1}. The use of the sigmoid function (represented as σ) scaled all entries of gate states i_t, f_t, o_t into the range between 0 to 1. Subsequently, the current information, $\tilde{c}_t$, is computed by x_t and h_{t-1} with a tanh activation function, consistent with the calculation in RNN. Most importantly, the current cell state is the linear combination of the weighted current and previous information, $i_t \odot \tilde{c}_t$ and $f_t \odot \tilde{c}_{t-1}$. As the computation

suggests, the input gate controls what new information is supposed to be kept in the cell state and the forget gate decides what old information the network would like to abandon. As the final step, the cell state is put through a tanh function, which pushes the output values into the range between -1 and 1. Then, the result is multiplied by the output gate state o_t that determines what information should be output [60, 62].

$$
\begin{aligned}
i_t &= \sigma(W^i x_t + U^i h_{t-1}) \\
f_t &= \sigma(W^f x_t + U^f h_{t-1}) \\
o_t &= \sigma(W^o x_t + U^o h_{t-1}) \\
\tilde{c}_t &= \tanh(W^c x_t + U^c h_{t-1}) \\
c_t &= i_t \odot \tilde{c}_t + f_t \odot \tilde{c}_{t-1} \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
\tag{24}
$$

Although the computation of LSTM is complicated, its way of storing and processing information is highly consistent with how human brains work [62]. As a result, LSTM has achieved huge success in challenging tasks of language processing, such as speech recognition and machine translation and, became the most popular choice of neural architectures for language modeling [60].

Because of the superior performance, researchers have created many variants of LSTM and employed them in language modeling. The AWD-LSTM (ASGD weight-dropped LSTM) is one of the most representative architectures among those variants [57]. AWD-LSTM adopts advanced regularization and optimization techniques to not only avoid overfitting caused by the high degree of complexity of LSTM but to achieve state-of-the-art results on benchmarks of language modeling [57]. Therefore, we chose the AWD-LSTM as the representative of a number of LSTM-based language models in our experiment.

It is worth mentioning that the bidirectional LSTM architecture is also used for building language models [58]. The bidirectional LSTM, as the name suggests, stands for a set of LSTM-based models that learn the conditional probability distribution of possible next words given contexts from both left and right sides. It is suggested that these models benefit from the future information being provided and have achieved significant performance in various of NLP tasks such as sentiment analysis even without fine-tuning [58]. Nevertheless, the bidirectional LSTM language models are not capable of generating text due to the lack of the right-hand context during generation. Here, we selected a representative bidirectional LSTM

architecture, contextual bi-LSTM [58], to perform our experiment.

3.3.4 Attention mechanism and the Transformer

In recent years, incredible progress has been made in language modeling, thanks to the introduction of the **attention mechanism**.

The concept of "attention" originated from the Seq2Seq (sequence-to-sequence) model which, as the name suggests, represents a group of models that transform the input sequence into another sequence [41]. The most intuitive example of the transformation is machine translation between multiple languages. A Seq2Seq model consists of an encoder that compresses the input sequence into a distributed context vector, and a decoder which transforms the context vector into the target sequence.

The earliest Seq2Seq models used RNNs for both encoder and decoder. According to what we learned about RNNs, the encoder will convert the input sequence into a fixed-length vector from which the decoder generates the output sequence. The limitation of the context vector is considered as a bottleneck that affects model performance. In 2015, Bahdanau et al. proposed the attention mechanism to deal with this issue [48].

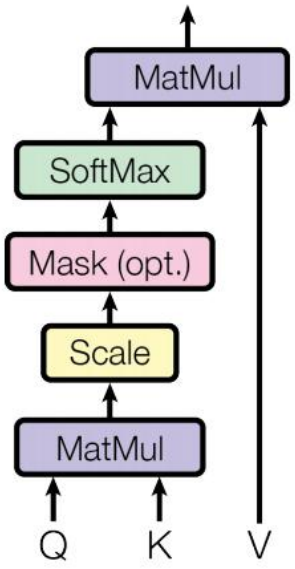

Figure 5: The attention mechanism: the values, V, are weighted and summed up by attention-weights that are obtained through a set of operations on the queries, Q, and the keys, K [64].

The attention mechanism is used to search for a set of positions in the input sequence where the most relevant information is concentrated. More specifically, a set of alignment scores describing how relevant each input hidden state and output element are will be learned while training the model. Fig.5 depicts the process flow of the attention mechanism, in which the attention operation is considered as a retrieval process [64, 65]. Based on the Figure 5, the operations are formulated by Equation 25, where d_k is the dimension of matrix K and (a) represents the vector of attention-weights. Through model training, the most likely parameters of the softmax layer and possible middle layers can be obtained and then be used to calculate a vector that aligns attention score for each value in V [65].

$$\begin{aligned} \boldsymbol{a} &= softmax(\frac{QK^T}{sqrt(d_k)}) \\ Attention(Q, K, V) &= \boldsymbol{a}V \end{aligned} \tag{25}$$

The attention mechanism makes better use of potential information of sequential data, and therefore achieved comparable performance on the task of English-to-French translation [48].

Subsequently, in 2017, a novel attention-based language model, the Transformer, was proposed by Ashish et al. [64]. This model dispenses with complex RNN architectures entirely and is based solely on the attention mechanism. This architecture surpasses complicated RNN-based models on machine translation tasks and requires significantly less time to train. The structure of the Transformer is demonstrated in Figure 6 [64].

As shown in Figure 6, both encoder and decoder of the Transformer are stacked by six identical layers. Each layer in the encoder has two sub-layers that perform multi-head attention and vector transformation respectively. A residual connection is employed in each of the sub-layers. Different from each encoder layer, a middle sub-layer that performs multi-head attention over the output of the encoder stack is inserted in each decoder layer.

It should be noted that the incredible performance achieved by Transformer benefits from the use of a new technique called **multi-head attention**. Different from the standard attention mechanism, multi-head attention repeatedly applies the scaled dot-product attention (the standard attention) on linear projections of Q, K, V multiple times. This allows the system to learn the attention weights from different representations of Q, K, V, which yields better performance. Figure 7 shows the structure of multi-head attention, and multi-head attention is written as Equation

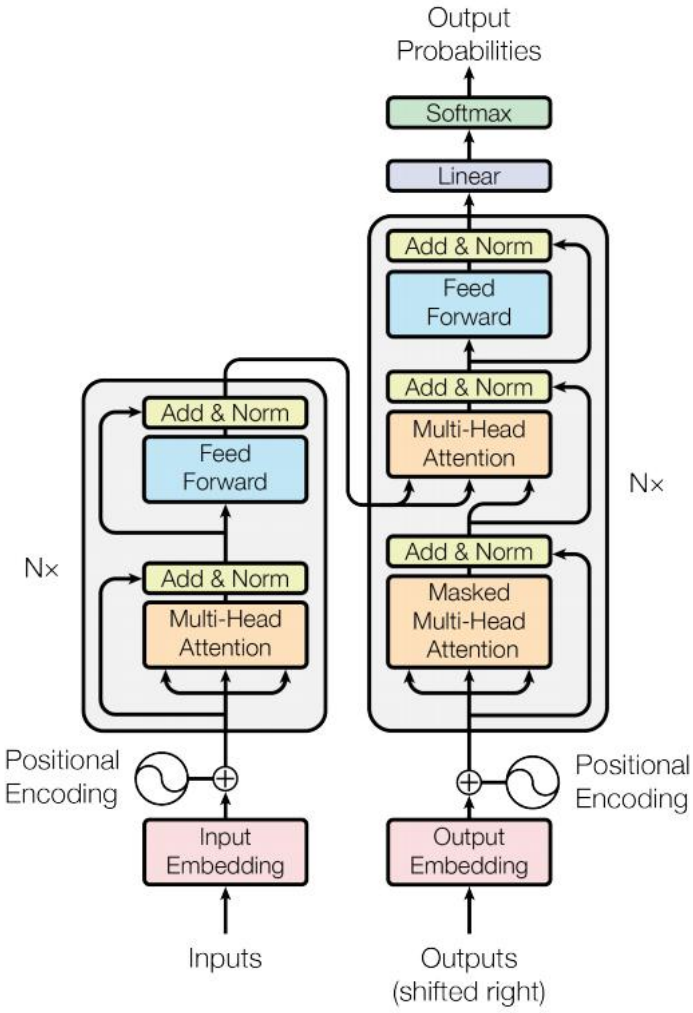

Figure 6: The structure of the Transformer: the stack on the left-side is the encoder while decoder is placed on the right [64].

26, where W_i^K, W_i^K, W_i^V are parameter matrices that are used to unify the dimensions of projections of matrices Q, K, V [64].

$$\begin{aligned} MultiHead(Q,K,V) &= Concat(head_1, ..., head_h)W^O \\ head_i &= Attention(QW_i^K, KW_i^K, VW_i^V) \end{aligned} \tag{26}$$

3.3.5 Pre-trained language models

It is worth mentioning that the great thing about the Transformer is not only its superior performance, but the fact that it has also led to the birth of two powerful pre-trained language models, BERT (2018, [17]) and GPT-2 (2019, [59]). BERT is a multi-layer bidirectional Transformer encoder. While GPT-2, the second version

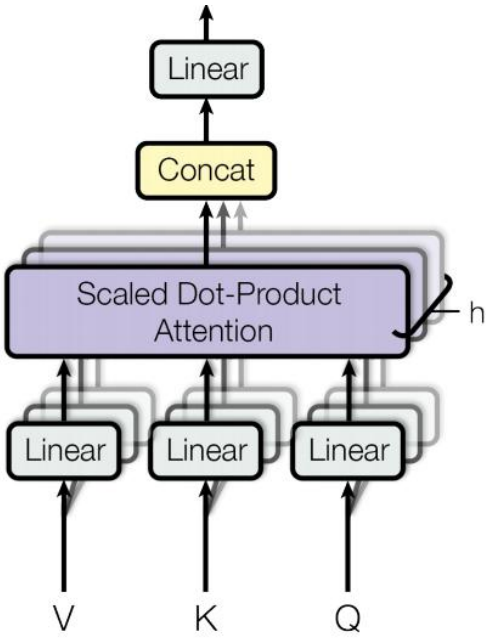

Figure 7: Multi-Head Attention consists of several parallel scaled dot-product attention layers [64].

of the OpenAI Generative Pre-training Transformer (2018, [66]), is a multi-layer Transformer decoder. Both BERT and GPT-2 are pre-trained on a giant collection of free online corpora. BERT was pre-trained on the BooksCorpus (800M words) and English Wikipedia (2,500M words), while GPT-2 was pre-trained on around 40GB of Internet text [59, 17]. They have achieved significant performance on multiple downstream NLP tasks, such as question answering, textual entailment and sentiment analysis by simply fine-tuning [17, 66, 59].

Basically, the language model pre-training aims to transfer the pre-trained language representations to downstream tasks in order to ease the difficulty of solving challenging NLP problems [17]. There are two strategies used by existing pre-trained language models: feature-based and fine-tuning. The most famous feature-based model is ELMo ((Embeddings from Language Models, 2018, [67]), which adopts pre-trained language representations on task-customized architectures and has significantly improved the model performance. The fine-tuning approaches, such as BERT and GPT-2, trained huge language models that can be reused by downstream tasks with little changes in model architectures [17]. Here, we chose to use the fine-tuning based model to get rid of the high level of complexity of finding optimal model architectures required by the feature-based approaches.

As the first fine-tuning based pre-trained model, OpenAI GPT [66] pre-trains a lan-

guage model on a giant collection of free online corpora at the first stage to capture the linguistic information. This step is performed in an unsupervised manner, which avoids the need for gathering a large labeled dataset. As the second step, the pre-trained language model is fine-tuned over a small task-customized dataset to ensure good performance on a specific task.

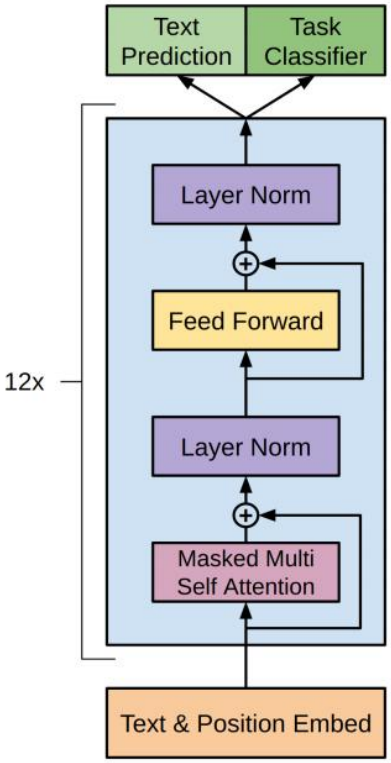

Figure 8: The architecture of the OpenAI GPT: the model contains a embedding layer, 12 blocks of the Transformer decoder, and a Softmax layer [66] .

The architecture of GPT is shown in Figure 8 [66]. The model takes embeddings of texts as input and outputs a distribution over the vocabulary after passing through 12 blocks of the Transformer decoder and a Softmax layer. As a supplement, the task-aware input transformations will be performed during the fine-tuning stage, in order to make sure the suitability of the pre-trained model for the target task [66].

Although it was demonstrated that GPT has incredible success in a wide range of benchmarks for natural language understanding, it may be limited by its uni-directional nature, which causes the ineffective utilization of the training data. Several months after the birth of GPT, a bi-directional pre-trained model, BERT, came out and beat its ancestor [17].

Instead of using the traditional method of training a language model, BERT adopts two auxiliary tasks: mask language modeling and next sentence prediction, to encourage better bi-directional prediction and sentence-level understanding. After training, the pre-trained BERT is fine-tuned by adding an extra output layer to suits the target task. BERT has achieved new state-of-the-art results on multiple downstream tasks, such as question answering and language inference [17].

Subsequently, a much larger and powerful pre-trained language model, GPT-2, was proposed by Radford et al [59]. GPT-2 inherits the architecture from GPT and is 10 times larger than GPT with 1.5 billion parameters. At the same time, it was pre-trained on a more than 10 times larger dataset to void overfitting. GPT-2 achieves state of the art results on 7 out of 8 benchmarks in different NLP tasks, including Penn Tree Bank, WikiText-2, Winograd Schema Challenge, and so on, in a zero-shot setting – without any fine-tuning. This property makes GPT-2 prominent among pre-trained language models [59].

Our choice of pre-trained language models finally settled on GPT-2 due to its potential of performing well on multiple language understanding tasks in a zero-shot setting.

4 Methodology

We used an unsupervised ranking-based method to explore if language models can be used to measure the text readability and text quality.

Our method is based on two hypotheses: i) language models trained on a well-written corpus will be less perplexed on articles that are simple and well-written, ii) the more predictable the words in the article are, the more readable and well-written the article is.

Based on the hypotheses, we proposed two metrics: average per word perplexity and average likelihood ratio, that we thought can measure how well a language model predicts text, to assess text readability and text quality. Here, we investigated the following questions:

1. Whether the metrics correlate well with readability scores or quality scores that are assigned by human raters.

2. What kind of language models can capture the characteristics of well-written

articles and therefore can be used to assess writing competence of articles.

To find the answer to the above questions, we performed multiple correlation experiments on various language models, such as bigram, LSTM, bidirectional LSTM and GPT-2. These language models were trained on the Guardian corpus, a British newspaper, that we thought is well-written. Further, we used three corpora, Newsela [4], OneStopEnglish [5] and Weebit [68] that are widely used in text readability studies, in order to verify the relation between text readability and the metrics. As for the study of text quality, we used the self-collected IELTS corpus and another ASAP corpus to measure the correlation between pre-assigned quality scores and the metrics.

In this chapter, the details of datasets and model architectures used in this thesis will be described in Chapter 4.1 and 4.2, respectively. Chapter 4.3 will demonstrate the intuitive derivation of the metrics, word perplexity and word ratio, followed by the introduction to the correlation coefficients that we used to evaluate our metrics in Chapter 4.4.

4.1 Dataset description

We used 6 corpora in total. The language models were trained using news articles from The Guardian, a British newspaper. We assume that these articles are well-written because they were written by journalists, who write for a living. Experiments for readability assessment were conducted on the Newsela, OneStopEnglish and WeeBit corpora, which are the three most popular datasets used for assessing readability and text simplification tasks. Besides, we adopted the IELTS corpus and the ASAP++ dataset to estimate the performance of text quality evaluation.

4.1.1 The Guardian Corpus

It is common for language models to be trained on English Wikipedia. However, as it has been criticized for the writing quality of Wikipedia articles [1], we instead decided to create our own corpus of well-written text. For this purpose, we downloaded news articles from The Guardian, a UK newspaper. Articles from newspapers are written by journalists, who are trained to write clearly and concisely. In addition, newspaper articles will have been checked by several editors for correctness and to achieve a consistent style prior to publication.

We crawled 20,000 news articles (May 12, 2019 - September 17, 2019) from four categories (news, culture, sport and lifestyle) of The Guardian's website. Each category contains 5,000 articles (downloaded September 18th, 2019). Each article was pre-processed by removing non-ascii characters and setting text to lowercase. Subsequently, we shuffled the ordering of the articles and split the corpus into training, validation and test sets in the proportions, 80%, 10% and 10%, respectively.

4.1.2 The Newsela Corpus

The Newsela corpus contains articles intended for students of various age groups and is widely used to assess text complexity and to evaluate the reading ability of students [4]. Here, we used the latest version of Newsela (from January 9th, 2016), which contains 10,786 articles, each of which is supplemented by up to five simplified versions of the same article. Simplified articles were written by experienced editors in order to make them easier to understand for students at different grade levels. We filtered out non-English articles and deleted those that did not have corresponding simplified versions. Of the remaining 9,565 articles, 28 articles had 3 simplified versions, 1,840 had 4 simplified versions, and the remaining 42 had 5 simplified versions.

4.1.3 The OneStopEnglish Corpus

The OneStopEnglish corpus was created by Vajjala et al. in 2016 and became publicly available in 2018 for automatic readability assessment and text simplification tasks [5]. The dataset contains 567 articles from 2013-2016 from the OneStopEnglish website. As an English language learning resources website, the OneStopEnglish website is mainly aimed at second language learners. The articles on the website are sourced from The Guardian and have been rewritten by experienced teachers into three versions, namely advanced, intermediate and elementary. This means that the corpus contains 3 different versions of 189 articles.

4.1.4 The WeeBit Corpus:

The WeeBit corpus was first used in the study of second language readability classification by Vajjala in 2012 [68]. The data was sourced from two websites: the Weekly Reader and BBC Bitesize, which are aimed at audiences of different ages. The articles from these two websites are merged and then divided into 5 categories

that correspond to the 7-8, 8-9, 9-10, 10-14, 14-16 age groups. Since there are multiple broken files in the dataset, we did a series of pre-processing steps, including removing empty and duplicate files, deleting files that containing anomalous text, and fixing coding issues. At the same time, in order to balance the number of articles in each category, we downsampled the group of age 14-16 which has the largest number of articles. In the end, we obtained a corpus that contains 3,096 articles.

4.1.5 The IELTS Corpus

We crawled 107 scored essays from the IELTS Academic test from IELT-Blog.com. The website provides various materials for students who aim to obtain good grades in IELTS Academic test. The IELTS academic test is a well-known English test among second language learners, especially among university students who desire to study in countries whose local language is English. The test consists of four parts: listening, speaking, reading and writing. As one of the hardest parts of the test, writing requires students to compose an essay of 250-350 words on a given topic. The grade ranges from 1 to 9, where band 9, the highest score, represents that the essay has a high degree of quality, while band 1 suggests that the essay uses very limited vocabulary and basically fails to convey any information.

Here, the 107 essays we used were scored from 5 to 9. The essay that has higher grade is considered to have better quality and be well-written. It should be noted that our dataset is imbalanced due to the limited access to scored essays of the IELTS test online.

4.1.6 The ASAP Dataset

The Automated Student Assessment Prize's dataset (ASAP) was first introduced by Shermis and Burstein in 2013 for automated essay evaluation [19]. After that, ASAP is widely used for evaluating the score of essays and have become a benchmark of automated essay scoring (AES).

The original ASAP dataset consists of 8 subsets that contains essays written over 8 prompts (given topics). The whole dataset includes 12,978 essays that contain 150 - 500 words. Each essay was double-graded by at least one and up to three proficient evaluators based on the overall quality of an essay. However, as our work aims to prove the relation between metrics we proposed and text quality which relates to a variety of aspects to consider, we assumed that the overall score is not enough for our use. Fortunately, in 2018, Mathias and Bhattacharyya [69] supplemented the

dataset by adding extra annotations (i.e. scores for different attributes of the essays, such as content, word choice, organization, sentence fluency, etc.). The new dataset is called ASAP++.

It should be noted that despite the fact that the ASAP++ dataset is considered as a whole, its 8 subsets adopt different sets of attributes and had different rating scales for each attribute. Therefore, we treated the ASAP++ as 8 separated datasets. For each set, we carefully selected an existing attribute that is most relevant to text quality to represent the quality scores of an essay given by human raters. More specifically, for sets 1, 2, and 8, we choose the attribute of sentence fluency while we use the attribute of style for set 7. And for sets 3 - 6, the attribute of narrativity is selected. The detailed information of each subset are listed in Table 1.

Table 1: The grade level, score scales, average length, and the number of essays in the ASAP++ dataset

Subset	**Grade level**	**Score range**	**Average length**	**#Essays**
set 1	8	1-6	350 words	1,785
set 2	10	1-6	350 words	1,800
set 3	10	0-3	150 words	1,726
set 4	10	0-3	150 words	1,772
set 5	8	0-4	150 words	1,805
set 6	10	0-4	150 words	1,800
set 7	7	0-3	250 words	1,730
set 8	10	1-6	650 words	918

4.2 Model description

To explore the impact of different language models on assessing text readability and quality, we used 4 language models that have different levels of complexity. Firstly, we used a bigram model as the baseline due to its simplexity and interpretability. Next, LSTM and bidirectional LSTM based language models that can capture semantic and syntactic information of language were selected. Last, but not least, we utilized the state-of-the-art GPT-2 language models to perform the experiments. We expect to show the difference of assessment ability due to distinctive competence of language models.

4.2.1 Bigram language model

The perplexity of the statistical language model is often presented as an important feature in papers that explored text readability in terms of simplicity [2, 10]. This seems to partially prove that the perplexity of SLMs is closely related to text readability. At the same time, with the consideration of model complexity, we decided to use a bigram model as it could capture simple context information with less sparsity and is also easy to implement. To handle the zero frequency problem brought by SLMs itself, we incorporated the add-0.5 smoothing to the bigram model.

4.2.2 AWD-LSTM

The LSTM architecture we used here is a variant of AWD-LSTM that was proposed by Merity et al. in 2017 [57]. AWD-LSTM stands for ASGD Weight-Dropped LSTM, which used DropConnect [53] and a non-monotonically triggered average stochastic gradient descent (SGD). These two techniques made the AWD-LSTM dominate in LSTM-based language modeling.

DropConnect [53] is an advanced regularization technique to avoid over-fitting of neural networks. Compared to the traditional regularization method Dropout, DropConnect instead sets randomly selected subsets of hidden to hidden weights to zero. It was empirically demonstrated that DropConnect is more efficient for regularization of RNN models [53].

On the other hand, non-monotonically triggered average SGD, a variant of the average SGD, represents the technique that switches SGD optimizer to average SGD only when the validation loss fails to drop in multiple cycles. The number of cycles is pre-defined and we used 5 cycles as its author suggested.

In our use, despite the fact that it was suggested that SGD optimization outperforms other techniques, such as Adam, momentum SGD, and RMSProp on the task of language modeling [70], we found that Adam optimization can help the model to converge to an optimal result faster. Therefore, we used a three-layer LSTM with weight-drop applied to perform language modeling. The model was trained in 300 epochs and converged after around 150 epochs.

4.2.3 Contextual bidirectional LSTM

The contextual BLSTM (cBLSTM for short) is based on the bi-directional LSTM (BLSTM) architecture [58]. As we mentioned before (section 3.3.3), BLSTM predicts each word using both left and right sides contexts, which provides more contextual information for language modeling. However, it was demonstrated that the traditional BLSTM will cause the problem of circular loops in sequence prediction. Take a simple sentence "<BOS> How are you <EOS>" as an example, at index 1, the forward LSTM will predict the word "are" while the backward LSTM will take "are" as the input. To solve the problem, the cBLSTM was proposed in 2017 for the sentiment analysis task [58].

Here, we used a one-layer cBLSTM that has 128 hidden units to build the language model. Even though we did not perform fine-tuning on the model, it obtained lower validation and test loss compared to the unidirectional LSTM we used. However, due to the higher level of model complexity, the model seems to overfit after 20 epochs.

4.2.4 GPT-2

Recently, language modeling has benefitted from the development of transfer learning [66, 17]. Among two most powerful pre-trained language models, BERT and GPT-2, we chose to use GPT-2 due to its state-of-the-art performance on generating coherent text [59].

Currently, there are three publicly available versions of GPT-2 called GPT2-small, GPT2-medium, and GPT2-large. As the name suggests, three versions have model sizes from small to large. Due to the limitation of computing resources, we only trained the smallest GPT-2 on the Guardian corpus in 20 epochs. That is to say, the GPT2-medium and GPT2-large have been directly used to perform the evaluation.

4.2.5 Training results

In this subsection, we describe the training result of language models. The number of parameters, validation and test perplexity of each model are shown in Table 2.

As we can see from Table 2, the simplest bigram model obtained an extremely high validation perplexity compared to other NNLMs. It suggestes that the bigram model did not capture the pattern of well-written text. Further, the fine-tuned WD-

Table 2: The number of parameters, validation and test perplexity of each language model

Model	**Parameters**	**Validation**	**Test**
Bigram model	2.19M	2024.41	1984.89
AWD-LSTM	95.39M	79.40	78.49
Contextual-biLSTM	158M	73.42	71.41
GPT-2 small	124M	19.99	19.75
GPT-2 medium	355M	23.08	22.80
GPT-2 large	774M	19.99	19.80

LSTM and cBLSTM had similar validation perplexity. We can conjecture that they might have similar performance on inferring well-written text that has the same distribution as the training set.

As for the three GPT-2 models, they largely outperformed the LSTM based models and the bigram model. Based on this, we infer that GPT-2 models might be the superior choice for capturing the pattern of well-written text. Despite the fact that GPT2-large was not trained on the Guardian corpus, it performs as well as the trained GPT2-small on the validation set. After training, GPT2-small performs better than the untrained GPT2-medium on both validation and test sets. This suggests that pre-trained larger scale GPT-2 model has strong generalization ability to infer language.

4.3 Word-level metrics

Imagine that you are reading an essay, it might be distracting when you see some unexpected words in the text. These words may be words that you are not familiar with or words that are grammatically incorrect or misspelled. Whatever they are, these unpredictable words have great potential to lead us to think that the article is difficult to read. On the contrary, when there are no words in the article that surprise you or make you feel uncomfortable, we often find the article highly readable and well-written. This intuitive feeling inspired us to assess text readability and quality by measuring how well a well-trained language model will predict each word appearing in the text. Here, we assume that the well-trained language model is able to determine if the use of words in a text is reasonable like human does.

Based on this, we proposed two metrics: average word perplexity and average likeli-

hood ratio, that largely depend on the probability of a word appearing in the text to assess text readability and quality. Note that both word perplexity and word ratio are word-level metrics that enable us to highlight issues for editing.

4.3.1 Word perplexity

The word perplexity is derived from Equation 5 by substitute the sequence length to 1. The formula of word perplexity is as follows.

$$WP(w) = \frac{1}{p(w)} \ , \tag{27}$$

where w represents the word of interest. However, the value of perplexity might be extremely large when the probability p is close to 0. Therefore, we represent the word perplexity in log format as WP_{log}:

$$WP_{log}(w) = -log(p(w)) \tag{28}$$

To measure the perplexity on the whole sequence or article that has N tokens, we calculate the mean of word perplexity as in Equation 29.

$$WP_{mean} = \frac{1}{N} \sum_{n=1}^{N} WP_{log}(w_n) \tag{29}$$

The word perplexity strongly depends on the word probability predicted by the language model. Therefore, word perplexity could demonstrate how likely a word appears in a specific position of a sequence. Note that because of the inverse in Equation 27, the higher the word probability, the lower the word perplexity. This suggests that the lower the average word perplexity is, the better the model will predict the article.

4.3.2 Likelihood ratio

The likelihood ratio, also called word ratio in this thesis, is the ratio of the word probability and the maximum probability of any word appearing in the position of interest. The formula is written as follows:

$$LR = \frac{p(w)}{\max_{1 \leq v \leq V} p(w_v)} \ , \tag{30}$$

where $p(w)$ represents the probability of the word w and V stands for the vocabulary size. Similarly, we use the mean of likelihood ratio of each word to evaluate the uncertainty of the whole text that has N tokens.

$$LR_{mean} = \frac{1}{N}\sum_{n=1}^{N}\frac{p(w_n)}{\max_{1\leq v\leq V} p(w_v)} \tag{31}$$

The likelihood ratio ranges from 0 to 1. When likelihood ratio equals to 1, the current word will be considered as the most likely word by the language model. As opposed to word perplexity, likelihood ratio is positively related to the predicted probability and, therefore, the higher the average likelihood ratio, the less perplexed the language model feels about the text.

4.4 Correlation coefficients

To test our assessment method, we measured the correlation coefficient between the human rater scores and the two metrics. There are two mostly used correlation measures, Pearson correlation coefficient (PCC) and Spearman's rank correlation coefficient (SRCC). The Pearson coefficient can measure the linear correlation between two continuous variables X and Y [71], while SRCC, a robust non-parametric measure, can identify the statistical dependence between the rankings of two variables [6]. Despite the fact that PCC has been widely used to evaluate the performance of readability assessment systems and AES systems [7, 6, 33], we argue that PCC is not suitable to measure the correlation as discrete variables are involved, such as grade levels or quality scores in our experiments. Also, it is shown that PCC is neither distributionally robust nor outlier resistant [72, 73]. As a result, we chose the Spearman coefficient as the correlation measure in our experiment.

4.4.1 Pearson correlation coefficient

Pearson correlation coefficient is a measure of linear correlation between two variables X and Y. It is defined as the covariance of two variables divided by the product of their standard deviation [71]:

$$\rho(X,Y) = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} \tag{32}$$

Here, σ represents the standard deviation while $\operatorname{cov}(X,Y)$ stands for the covariance of variables X and Y. According to the Cauchy–Schwarz inequality, $\rho_{X,Y}$ has the

value between -1 and +1, where -1 refers to total negative correlation, +1 represents the total positive linear correlation, while 0 stands for no linear correlation [73].

4.4.2 Spearman's rank correlation coefficient

Spearman's rank correlation coefficient, a robust non-parametric metric, is defined as the Pearson coefficient between rank variables. Unlike PCC that evaluate the linear relationship between variables, SRCC can instead measure the monotonicity between two variables. For variables X and Y, we first need to obtain their corresponding ranking variables X' and Y', in order to calculate the SRCC. Subsequently, we could compute the SRCC using the PCC between the variables:

$$r_s(X, Y) = \rho(X', Y') \ . \tag{33}$$

SRCC has ranges from -1 and +1. The Spearman correlation of -1 (or +1) occurs only when each variable is a perfect monotonically decreasing (or increasing) function of the other, while 0 coefficient suggests that there is no monotonicity of the relationship between two variables.

5 Experiments

We performed correlation experiments on both tasks to evaluate how much our metrics correlate with scores rated by human annotators. In this chapter, we depict the experimental details for both tasks, i.e., readability assessment and text quality evaluation in subsections 5.1 and 5.2, respectively. Subsequently, more detailed discussion on the comparison between metrics are presented in 5.3, followed by a case study to show the effectiveness of our methods on text quality evaluation.

5.1 Correlation experiments on text readability

The basic idea of the experiment is to directly evaluate the correlation between our metrics and the ground-truth scores (i.e. scores given by human raters). In the datasets used for readability evaluation, ground-truth scores are usually the grade levels of texts. And the readability scores we obtained are computed based on the outputs of language models, according to Equations 27-31. What we expected are strong positive and negative correlations between the ground-truth scores and readability scores of word perplexity and likelihood ratio.

5.1.1 Experimental setup

We evaluated the effectiveness of our metrics, word perplexity and likelihood ratio, using the Newsela, OneStopEnglish, and WeeBit. We employed four different architectures of language models, the bigram model, AWD-LSTM, contextual bidirectional LSTM, and GPT-2, in the experiments. These language models were trained on the Guardian corpus that contains well-written articles. By doing this, we believe that the trained language models can capture the characteristics of highly readable articles and, therefore, can be used to evaluate text readability.

We used SRCC to measure the correlation between our metrics and grade levels. Larger absolute values of correlation coefficients indicate stronger correlations. As a comparison, we also computed the correlations between grade levels and multiple traditional readability measures, including FKGL, GFI, ARI, SMOG and Dale-Chall score. The average sentence length and article length are used as baselines.

In addition, to further explore the impact of model size and training on the readability assessment method, we included four different versions of GPT-2: GPT-2 train, GPT-2 small, GPT-2 medium and GPT-2 large. Here, GPT-2 small, medium and large represent three released architectures of GPT-2, while GPT-2 train refers to a GPT-2 small trained on the Guardian corpus.

5.1.2 Results

The results of the correlation experiments for the task of readability evaluation are presented in Table 3.

The metrics we proposed achieved lower correlation than traditional readability measures. The highest correlation achieved by our method on the Newsela corpus is $\boldsymbol{\rho} = 0.2567$ (by GPT2-small), which is much lower than 0.9459 obtained by the traditional measure – GFI. The best coefficients achieved by the proposed measures on the other two datasets are slightly higher. On the Weebit corpus, the best result of our method is achieved by GPT2-large ($\boldsymbol{\rho} = 0.3527$), while the GFI ended up with much better result ($\boldsymbol{\rho} = 0.6195$). Similarly, average perplexity measure of GPT2-small yielded $\boldsymbol{\rho} = 0.3241$ on the OneStopEnglish corpus, while 0.6802 is achieved by one of the baseline metrics – the essay length, as the highest.

Further, we found that both perplexity-based and ratio-based measures perform inconsistently over the three corpora. On the one hand, the perplexity-based measures have negative correlations with labeled grade levels. The bigram perplexity

Table 3: Spearman's rank correlation coefficients between grade levels and word perplexity, likelihood ratio, traditional readability measures.

Model	Metric	Newsela	Weebit	OneStop
Bigram model	Word perplexity	0.2346	-0.0858	0.2544
AWD-LSTM		0.0919	0.0379	0.2386
Contextual-biLSTM		-0.4355	-0.1059	0.0725
GPT-2 train		0.2071	0.1853	0.2759
GPT-2 small		**0.2567**	0.3294	**0.3241**
GPT-2 medium		0.2085	0.3428	0.2633
GPT-2 large		0.2276	**0.3527**	0.2867
Bigram model	Likelihood ratio	0.04816	-0.0428	0.0137
AWD-LSTM		-0.2685	-0.1141	-0.2346
Contextual-biLSTM		0.2670	0.2129	0.0608
GPT-2 train		-0.1202	-0.2128	-0.1812
GPT-2 small		**-0.1787**	-0.3257	**-0.2589**
GPT-2 medium		-0.1401	-0.3293	-0.1835
GPT-2 large		-0.1542	**-0.3402**	-0.2091
Traditional measures	Flesch-Kincaid grade	0.9310	0.6009	0.5721
	Gunning Fog	**0.9459**	**0.6195**	0.6556
	Automated index	0.9367	0.4847	0.5880
	Smog index	0.8864	0.4598	0.5195
	Dale-Chall score	0.8884	0.5029	0.6516
	Average sent. length	0.9393	0.5486	0.5970
	Essay length	0.6979	0.0599	**0.6802**

measure obtained $\boldsymbol{\rho}$ of -0.0858 on the Weebit corpus, and the C-biLSTM perplexity scores are negatively correlated with readability scores on both Newsela and Weebit corpora. In addition, the AWD-LSTM perplexity measure achieved the lowest correlation on both Newsela and Weebit datasets ($\boldsymbol{\rho}$ of 0.0919 and 0.0370, respectively). On the other hand, ratio-based measures similarly achieved positive correlations. C-biLSTM ratio measure is positively correlated with readability scores on all datasets, while the bigram measure obtained positive correlations on both Newsela and Weebit corpora.

However, we observed that four GPT-2 measures perform consistently across different corpora. These four measures achieved 0.20 – 0.35 $\boldsymbol{\rho}$ values with respect to

the average word perplexity and -0.12 – -0.34 to the mean likelihood ratio. As the GPT-2 has a well-designed architecture and is pre-trained on a large collection of online texts, it is much more powerful compared to other language models in terms of language understanding and inference. Other language models are only trained on the Guardian corpus,which may be too small for the models to capture the characteristics of well-written texts, and therefore failed the task of readability assessment. Also, the Guardian articles are aimed at adults, so language models trained on it may not be able to identify the grade levels of articles targeting students. Despite the low correlation coefficients achieved by four GPT-2 measures, we think that the consistency between them supports our assumption to some extent.

In fact, we found that the grade levels in three corpora, particularly in the Newsela corpus, strongly correlate with traditional readability measures. We suspect that these corpora were built under the guidance of traditional readability formulas by replacing complicated words and using short sentences. This somehow disrupts the coherence of articles in the dataset. It is well noted that the language model can capture not only shallow-level lexical information but deeper linguistic features such as discourse cohesion of texts, which suggests that all these features are considered by the language model when determining how predictable the next word is given the previous context. Therefore, the well-trained language model is very likely to be more perplexed on articles from lower grade levels due to their poor coherence (a cased study shows this in Section 5.3.3). This yields weak correlations between our metrics and grade levels of articles. Under such consideration, we argue that our metrics, word perplexity and likelihood ratio, are not viable indicators for readability assessment in terms of comprehension.

5.2 Correlation experiments on text quality

In this section, we describe the correlation experiments for the task of text quality evaluation. Similar to readability assessment, we compute the word perplexity and likelihood ratio of each essay using language models based on Equations 27-31, and we then measure the correlations between two metrics and quality scores graded by human raters. As we hypothesize that language models trained on well-written articles are less perplexed with well-written essays, we expect that the word perplexity and annotated quality scores will show a strongly negative correlation. And in contrast, the likelihood ratio is supposed to positively correlate with ground-truth scores.

5.2.1 Experimental setup

We adopt the same set of language models used for assessing text readability to calculate two metrics on two corpora, the IELTS corpus and ASAP++ (see details in Section 4.1). Then, we measure how our metrics correlate with annotated quality scores using SRCC. As the IELTS corpus contains only 104 examples, we verify the significance of obtained correlation coefficients by checking if the P-value is less than 0.05. We selected the article length as the baseline here for the performance comparison.

5.2.2 Results

Table 4 shows the results on the IELTs corpus. And results of correlation experiments on the ASAP++ dataset are presented in Tables 5 and 6.

Table 4: Spearman's rank correlation coefficients between text quality scores and word perplexity, likelihood ratio on the IELTs corpus.

Model/Metric	**Word Perplexity**	**Likelihood ratio**
Bigram model	0.4819	0.1163
AWD-LSTM	-0.0213	0.08225
Contextual-biLSTM	0.1177	-0.1059
GPT-2 train	-0.5680	0.4771
GPT-2 small	-0.5530	0.4550
GPT-2 medium	**-0.5956**	0.5000
GPT-2 large	-0.5844	**0.5118**
Article length	0.3764	

As we can see, the correlation coefficients of all GPT-2 measures achieve high correlations over almost all datasets, while other model-based measures perform inconsistently across different corpora. On the IELTs corpus, the highest correlation is $\boldsymbol{\rho}$ of 0.5950 that was achieved by GPT-2 medium perplexity-based measure. In the following four sets of the ASAP++: 2, 3, 5, 6, the best correlation coefficients are slightly lower but still over 0.5. Similarly, the GPT-2 word perplexity measures attained good results on ASAP++ 1 and 4 ($\boldsymbol{\rho}$ of 0.4190 and 0.4868, respectively). The worst performance is obtained on the remaining two sets ASAP++ 7 and 8, which have $\boldsymbol{\rho}$ of 0.3360 and 0.2295, respectively. Overall, GPT-2 based measures achieved convincing results on 7 out of total 9 datasets, which suggests that the

proposed metrics are reasonable indicators of text quality.

Table 5: Spearman's rank correlation coefficients between quality scores and word perplexity, likelihood ratio on ASAP++ subsets 1-4. Score with a '$*$' mark represent that the correlation is not statistically significant.

Model	**Metric**	**set 1**	**set 2**	**set 3**	**set 4**
Bigram model	Word perplexity	0.0763	-0.0707	0.2064	0.3101
AWD-LSTM		-0.2347	-0.3784	-0.1608	0.0110*
Contextual-biLSTM		-0.0727	-0.1067	-0.1493	0.0269*
GPT-2 train		-0.4140	**-0.5037**	-0.4485	-0.5299
GPT-2 small		-0.4006	-0.4736	-0.4550	-0.5472
GPT-2 medium		**-0.4190**	-0.4807	-0.4787	-0.5683
GPT-2 large		-0.4074	-0.4639	**-0.4868**	**-0.5760**
Bigram model	Likelihood ratio	0.2703	0.3172	-0.0203*	0.0387*
AWD-LSTM		0.3604	0.4745	0.1175	0.1629
Contextual-biLSTM		0.0673	-0.0252*	0.1831	0.2609
GPT-2 train		0.3739	0.4256	0.3433	0.5365
GPT-2 small		0.3747	0.4094	0.4123	0.5857
GPT-2 medium		**0.3912**	**0.4317**	0.4316	**0.5881**
GPT-2 large		0.3811	0.4202	**0.4395**	0.5856
/	Article length	0.6026	0.4797	0.6725	0.6943

Similar to the analysis for the readability experiments, we infer that the poor performance of other language models is caused by either the training dataset or the model architectures we used is not appropriate for the task of interest.

We compared the correlation coefficients achieved by our metrics with the baseline metric – article length. However, except for the IELTs corpus and ASAP++ set 2, the baseline metric exceeds proposed metrics by 8.3% to 68.6%. We think this is due to the inherent shortcomings of our metrics and the uncertainty of the language model itself. As stated in Section 5.2.3, word perplexity only uses single element from the predicted probability distribution, making it less representative for the whole distribution. The likelihood ratio is invalid in some specific contexts. These deficiencies are very likely to affect the accuracy for text quality evaluation. Despite this, our work fully demonstrates the potential of using distributed representations output by a well-trained language model to identify the text quality. We believe that the performance of our method can be further improved with more powerful

language models and well-designed metrics.

Table 6: Spearman's rank correlation coefficients between quality scores and word perplexity, likelihood ratio on ASAP++ subsets 5-8. Score with a '$*$' mark represent that the correlation is not statistically significant.

Model	**Metric**	**set 5**	**set 6**	**set 7**	**set 8**
Bigram model	Word perplexity	0.2891	0.1758	0.2363	0.1668
AWD-LSTM		-0.1049	-0.1505	0.0694	0.0035*
Contextual-biLSTM		-0.1599	-0.0467	0.0623*	0.1499
GPT-2 train		-0.5006	-0.5127	**-0.3360**	**-0.2065**
GPT-2 small		-0.4953	-0.5227	-0.3186	-0.1954
GPT-2 medium		-0.5166	-0.5390	-0.3239	-0.2048
GPT-2 large		**-0.5220**	**-0.5434**	-0.3273	-0.1831
Bigram model	Likelihood ratio	-0.0815	0.0793	-0.0521	0.2172
AWD-LSTM		0.1177	0.2130	-0.0114*	0.1815
Contextual-biLSTM		0.1559	0.1352	0.1858	0.0074*
GPT-2 train		0.4102	0.4037	0.2724	0.2039
GPT-2 small		0.4337	0.4358	0.2848	0.2156
GPT-2 medium		0.4596	0.4571	**0.2949**	**0.2295**
GPT-2 large		**0.4656**	**0.4642**	0.2938	0.2254
/	Article length	0.6827	0.5887	0.5666	0.3747

We further explore the impact of model capacity and training by comparing the results obtained by four GPT-2 based models on 9 datasets. For both perplexity-based and ratio-based measures, the correlations of GPT-2 train measures do not increase significantly compared to that of GPT-2 small measures. Instead, correlation coefficients decline on three datasets after training. Therefore, it is difficult to say that using language models that are trained on well-written articles will improve the evaluation accuracy. At the same time, we observe that GPT-2 medium measures achieved better correlations than GPT-2 small measures on all datasets, while GPT-2 large measures are sometimes even weaker than GPT-2 small measures. Despite this, we still believe that larger capacity of language models does improve the evaluation performance. It is suggested that three versions of GPT-2 are underfitting given their large architectures. Therefore, the degraded performance of GPT-2 large is most likely to be caused by its higher degree of underfitting, in our opinion.

To summarize, it is demonstrated that proposed measures with GPT-2 are strongly

correlated with text quality scores rated by human annotators. Despite the fact that not all correlations are significantly strong, our work suggests that distributed representations output by pre-trained language models have the capacity for determining the text quality in a direct form. By showing that GPT-2 measures performs similarly well on 7 out of 9 datasets in a zero-shot setting, we demonstrate the high degree of transferability of our method in text quality evaluation.

5.3 Discussion

In this chapter, we further discuss the difference between the proposed metrics in Section 5.3.1. A case study of text quality evaluation is performed in Section 5.3.2.

5.3.1 Word perplexity VS likelihood ratio

Two metrics, word perplexity and likelihood ratio, are proposed to measure the concordance of article text with predictions made using neural language models. The motivation for using word perplexity originated from the fact that perplexity is widely used to measure the fitness between a language model and its input texts. Referring to the definition of perplexity, word perplexity is directly formulated as the inverse of the probability of the target word predicted by a language model. However, it may not be a good summary of the whole distribution that the language model outputs as it only uses one element from the distribution vector. More specifically, word perplexity cannot distinguish between distributions that have the same probability for target words.

Inspired by this, we proposed the likelihood ratio that is the ratio of word probability and maximum probability in the distribution that a language model outputs as a more accurate indicator. Intuitively, we think that the maximum value indicates the predictive ability of a language model. By dividing the maximum value of the whole distribution, the metric reduces the bias brought by the language model itself to some extent.

We infer that two metrics are intrinsically related. Our experiments verified this assumption. It was demonstrated that two metrics have a strong negative linear relationship on multiple datasets we used. Without loss of generality, the scatter plot of word perplexity and likelihood ratio of articles from the ASAP++ dataset is shown in Figure 9.

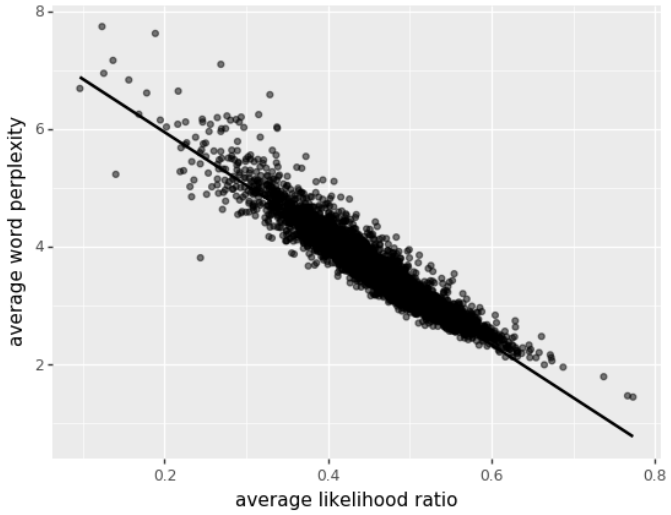

Figure 9: Relationship between word perplexity and likelihood ratio: x-axis and y-axis represent the average likelihood ratio and average word perplexity of each article in ASAP++. The straight line shows the linear correlation between two variables.

Despite that we thought likelihood ratio is a better metric, our experiments shows that perplexity-based measures usually achieved higher correlations than ratio-based measures. This may be because we did not consider the uncertainty of language models when using likelihood ratio to measure how well the article fits the language model. Consider the situation that a language model obtains equally low probability for each possible next word at a certain position, it is intuitive to say that the language model is bad at predicting the next word in this context and the predicted probability distribution is actually not trustworthy. Under this circumstance, the likelihood ratio reaches its maximum of 1. However, it is obvious that this high ratio value is not able to indicate how predictable the target word is and therefore is invalid in this case.

5.3.2 A case study of text quality evaluation

To intuitively show the effectiveness of our metrics, we selected two short articles from Simple English Wikipedia for a case study of text quality evaluation. Simple English Wikipedia, as the name suggests, is a simple version of the regular Wikipedia. Its articles are written at a basic level of English and aim at both children and English learners. Two samples we chose are shown in Figure 10. We refer two articles as 'Ronald article' and 'Grendel article', respectively, based on their

topics.

> The Ronald Reagan Federal Building and Courthouse is a federal building and courthouse named after Ronald Reagan . It is located at 228 Walnut Street in Harrisburg , Pennsylvania . It is twelve - stories tall . The building was built in 1966 . It is owned by the General Services Administration . It was officially renamed on March 9 , 2004 .
>
> Grendel is one of the three antagonists in the poem " Beowulf " . Grendel is usually taken to be some kind of monster , though this is the subject of scholarly debate . In the poem , Grendel is feared by all but Beowulf , who kills him and his mother at the end of the poem . He is the descendant of the biblical murderer Cain .

Figure 10: Two samples selected from Simple English Wikipedia. The first sample talks about the 'Ronald Reagan Federal Building and Courthouse', and the second article has the topic of 'Grendel'.

The articles we selected are good examples for demonstration due to their simplicity and shortness. Moreover, two articles are of similar length (62 tokens vs 68 tokens) and have a similar number of name entities (10 vs 8). This largely eliminates the effect of other factors on article quality evaluation and, therefore, make our analysis more accurate.

Intuitively, the Ronald article consists of multiple short sentences and is considered to be badly written by a native English speaker. In contrast, the Grendel article is well-structured and considered to be well-written. For all language models, the Grendel article obtained lower word perplexity, as shown in Table 7.

Table 7: Average word perplexity of two articles

Model	**Ronald article**	**Grendel article**
Bigram model	8.4378	7.1046
AWD-LSTM	6.0633	5.3087
Contextual-biLSTM	12.5675	12.4382
GPT-2 large	2.7860	2.6157

The result indicates that all language models we used feel less perplexed about Grendel article and consider it has better narrativity than Ronald article. This is in line with our intuition. As a comparison, we also measure the grade level of two samples using GFI, without loss of generality. The resulted grade levels for Ronald article and Grendel article are 8.04 and 10.0, respectively. Indeed, it is reasonable that the grade level of Ronald article is lower than that of the Grendel article, as it contains only simple and short sentences. However, this result is exactly the opposite

to our quality evaluation result, which suggests that our metric is somehow difficult to identify the simplicity of an article.

6 Conclusion and future work

In this thesis, we aimed to find an efficient way to evaluate whether an essay is readable and well-written, with the help of language models. Our work was based on the intuitive hypothesis that language models trained on a well-written corpus will be less perplexed on articles that are simple and well-written and can, therefore, determine how readable and well-written an article is. According to this, we proposed two word-level metrics, word perplexity and likelihood ratio that can identify how well articles fit a language model trained on well-written text, as the measurements of text readability and quality. We also explored the suitability of different kinds of language models, including a statistical bigram model, unidirectional and bidirectional LSTM models as well as attention-based GPT-2, for the task of text evaluation. To investigate the effectiveness of the metrics we proposed, we used Spearman's rank correlation coefficient to measure how our metrics correlate with scores annotated by human raters on multiple datasets that are widely used for readability assessment and text quality evaluation.

We tested our metrics on two tasks, i.e. readability assessment and text quality evaluation, separately. For assessing readability in terms of simplexity, we found that the performance of the bigram model, LSTM and bidirectional LSTM varies dramatically over the three corpora: Newsela, Weebit and OneStopEnglish. The inconsistent results indicated that these models did not fit the task well. Despite the fact that four versions of GPT-2 perform consistently on these three sets, the correlations between both of our metrics and real grade levels are quite weak. We also noticed that our metrics lag far behind traditional readability measures. These facts suggest that our method is not a good fit for the task of readability estimation.

Conversely, the same method achieved much better performance in the evaluation of text quality. Although other types of language models still failed, all 4 versions of GPT-2 achieve fairly high correlations on 7 out of 9 datasets (including the IELTS set and 8 subsets of ASAP++). The absolute values of coefficients range from 0.4 to 0.6. We demonstrated that even though both metrics obtained by GPT-2 are suitable measures for the text quality, the word perplexity exceeds the likelihood ratio in most cases.

In future work, we think there are three major issues that require more explorations. Firstly, our experiments shows that GPT-2 is better compared to LSTM models and bigram models on both tasks. And we are interested in figuring out why this happened. Secondly, we want to further investigate the impact of name entities for the task of text quality evaluation. This may help us to improve the performance of our method. Finally, as the goal of the project, we would like to build an editing assistant system using the proposed two word-level metrics, enabling highlight issues directly to editors in the future.

References

1 M. Hu, E.-P. Lim, A. Sun, H. W. Lauw, and B.-Q. Vuong, "Measuring article quality in wikipedia: models and evaluation," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pp. 243–252, 2007.

2 L. Si and J. Callan, "A statistical model for scientific readability," in *Conference on Information and Knowledge Management*, vol. 1, pp. 574–576, 2001.

3 L. Feng, N. Elhadad, and M. Huenerfauth, "Cognitively motivated features for readability assessment," in *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 229–237, Association for Computational Linguistics, 2009.

4 W. Xu, C. Callison-Burch, and C. Napoles, "Problems in current text simplification research: New data can help," *Transactions of the Association for Computational Linguistics*, vol. 3, pp. 283–297, 2015.

5 S. Vajjala and I. Lučić, "OneStopEnglish corpus: A new corpus for automatic readability assessment and text simplification," in *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pp. 297–304, Association for Computational Linguistics, 2018.

6 D. Pérez-Marín, I. Pascual-Nieto, and P. Rodríguez, "Computer-assisted assessment of free-text answers," *The Knowledge Engineering Review*, vol. 24, no. 4, pp. 353–374, 2009.

7 M. Martinc, S. Pollak, and M. R. Šikonja, "Supervised and unsupervised neural approaches to text readability," *arXiv preprint arXiv:1907.11779*, 2019.

8 R. Gunning, *Technique of clear writing.* McGraw-Hill, 1968.

9 J. P. Kincaid, R. P. Fishburne Jr, R. L. Rogers, and B. S. Chissom, *Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel.* Research Branch report, Institute for Simulation and Training, University of Central Florida, 1975.

10 S. E. Petersen and M. Ostendorf, "A machine learning approach to reading level assessment," *Computer speech & language*, vol. 23, no. 1, pp. 89–106, 2009.

11 M. Mesgar and M. Strube, "A neural local coherence model for text quality assessment," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4328–4339, 2018.

12 E. Dale and J. S. Chall, "A formula for predicting readability: Instructions," *Educational research bulletin*, pp. 37–54, 1948.

13 G. H. Mc Laughlin, "Smog grading-a new readability formula," *Journal of reading*, vol. 12, no. 8, pp. 639–646, 1969.

14 S. Zhou, H. Jeong, and P. A. Green, "How consistent are the best-known readability equations in estimating the readability of design standards?," *IEEE Transactions on Professional Communication*, vol. 60, no. 1, pp. 97–111, 2017.

15 I. M. Azpiazu and M. S. Pera, "Multiattentive recurrent neural network architecture for multilingual readability assessment," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 421–436, 2019.

16 S. E. Schwarm and M. Ostendorf, "Reading level assessment using support vector machines and statistical language models," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pp. 523–530, Association for Computational Linguistics, 2005.

17 J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

18 E. B. Page, "Grading essays by computer: Progress report.," in *Proceedings of the invitational Conference on Testing Problems*, 1967.

19 M. D. Shermis and J. Burstein, *Handbook of automated essay evaluation: Current applications and new directions.* Routledge, 2013.

20 F. Dong, Y. Zhang, and J. Yang, "Attention-based recurrent convolutional neural network for automatic essay scoring," in *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pp. 153–162, 2017.

21 M. D. Shermis and J. C. Burstein, *Automated essay scoring: A cross-disciplinary perspective*. Routledge, 2003.

22 P. W. Foltz, W. Kintsch, and T. K. Landauer, "The measurement of textual coherence with latent semantic analysis," *Discourse processes*, vol. 25, no. 2-3, pp. 285–307, 1998.

23 J. Burstein, K. Kukich, S. Wolff, C. Lu, M. Chodorow, L. Braden-Harder, and M. D. Harris, "Automated scoring using a hybrid feature identification technique," in *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pp. 206–210, Association for Computational Linguistics, 1998.

24 E. B. Page, J. P. Poggio, and T. Z. Keith, *Computer Analysis of Student Essays: Finding Trait Differences in Student Profile*. ERIC, 1997.

25 M. D. Shermis, H. R. Mzumara, J. Olson, and S. Harrington, "On-line grading of student essays: Peg goes on the world wide web," *Assessment & Evaluation in Higher Education*, vol. 26, no. 3, pp. 247–259, 2001.

26 L. Rudner and P. Gagne, "An overview of three approaches to scoring written essays by computer. eric digest.," *Practical Assessment, Research, and Evaluation:Vol.7*, 2001.

27 L. S. Larkey, "Automatic essay grading using text categorization techniques," in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, vol. 98, pp. 90–95, 1998.

28 L. M. Rudner and T. Liang, "Automated essay scoring using bayes' theorem," *The Journal of Technology, Learning and Assessment*, vol. 1, no. 2, 2002.

29 D. Marcu, "The rhetorical parsing of unrestricted texts: A surface-based approach," *Computational linguistics*, vol. 26, no. 3, pp. 395–448, 2000.

30 P. D. Turney, "Mining the web for synonyms: Pmi-ir versus lsa on toefl," in *European conference on machine learning*, pp. 491–502, Springer, 2001.

31 J. Burstein, C. Leacock, and R. Swartz, "Automated evaluation of essays and short answers," in *Proeedings of the 5th CAA Conference*, © Loughborough University, 2001.

32 Y. Attali and J. Burstein, "Automated essay scoring with e-rater® v. 2," *The Journal of Technology, Learning and Assessment*, vol. 4, no. 3, 2006.

33 H. Yannakoudakis, T. Briscoe, and B. Medlock, "A new dataset and method for automatically grading esol texts," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pp. 180–189, Association for Computational Linguistics, 2011.

34 H. Chen and B. He, "Automated essay scoring by maximizing human-machine agreement," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1741–1752, 2013.

35 D. Alikaniotis, H. Yannakoudakis, and M. Rei, "Automatic text scoring using neural networks," *arXiv preprint arXiv:1606.04289*, 2016.

36 K. Taghipour and H. T. Ng, "A neural approach to automated essay scoring," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1882–1891, 2016.

37 F. Dong and Y. Zhang, "Automatic features for essay scoring–an empirical study," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1072–1077, 2016.

38 D. M. Christopher, R. Prabhakar, and S. Hinrich, "Introduction to information retrieval," *An Introduction To Information Retrieval*, vol. 151, no. 177, p. 5, 2008.

39 C. D. Manning, C. D. Manning, and H. Schütze, *Foundations of statistical natural language processing*. MIT press, 1999.

40 E. Sapir, *An introduction to the study of speech*. Citeseer, 1921.

41 D. Jurafsky and J. H. Martin, *Speech and Language Processing (2Nd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2009.

42 R. Rosenfeld, "Two decades of statistical language modeling: Where do we go from here?," *Proceedings of the IEEE*, vol. 88, no. 8, pp. 1270–1278, 2000.

43 R. Kuhn and R. De Mori, "A cache-based natural language model for speech recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 12, no. 6, pp. 570–583, 1990.

44 P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin, "A statistical approach to machine translation," *Computational linguistics*, vol. 16, no. 2, pp. 79–85, 1990.

45 F. Jelinek, B. Merialdo, S. Roukos, and M. Strauss, "A dynamic language model for speech recognition," in *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19-22, 1991*, 1991.

46 T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean, "Large language models in machine translation," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 858–867, 2007.

47 K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

48 D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

49 A. Talman, A. Yli-Jyrä, and J. Tiedemann, "Sentence embeddings in nli with iterative refinement encoders," *arXiv preprint arXiv:1808.08762*, 2018.

50 Y. Wang, A. Mohamed, D. Le, C. Liu, A. Xiao, J. Mahadeokar, H. Huang, A. Tjandra, X. Zhang, F. Zhang, *et al.*, "Transformer-based acoustic modeling for hybrid speech recognition," *arXiv preprint arXiv:1910.09799*, 2019.

51 Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

52 Y. Goldberg, "Neural network methods for natural language processing," *Synthesis Lectures on Human Language Technologies*, vol. 10, no. 1, pp. 1–309, 2017.

53 L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *International conference on machine learning*, pp. 1058–1066, 2013.

54 W. A. Gale and K. W. Church, "What's wrong with adding one?," *Corpus-Based Research into Language: In honour of Jan Aarts*, pp. 189–200, 1994.

55 Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 933–941, JMLR. org, 2017.

56 T. Mikolov, A. Deoras, S. Kombrink, L. Burget, and J. Černockỳ, "Empirical evaluation and combination of advanced language modeling techniques," in *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

57 S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing lstm language models," *arXiv preprint arXiv:1708.02182*, 2017.

58 A. Mousa and B. Schuller, "Contextual bidirectional long short-term memory recurrent neural network language models: A generative approach to sentiment analysis," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pp. 1023–1032, 2017.

59 A. l. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, 2019.

60 Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*. Nature Publishing Group, 2015.

61 A. Karpathy, "Neural network 1 - convolutional neural networks for visual recognition." `http://cs231n.github.io/neural-networks-1/`, 2019.

62 C. Olah, "Understanding lstm networks." `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 2015.

63 Y. Bengio, P. Simard, P. Frasconi, *et al.*, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

64 A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

65 M. Allard, "What is a transformer?." https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04, 2019.

66 A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

67 M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018.

68 S. Vajjala and D. Meurers, "On improving the accuracy of readability classification using insights from second language acquisition," in *Proceedings of the seventh workshop on building educational applications using NLP*, pp. 163–173, Association for Computational Linguistics, 2012.

69 S. Mathias and P. Bhattacharyya, "ASAP++: Enriching the ASAP automated essay grading dataset with essay attribute scores," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

70 A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, "The marginal value of adaptive gradient methods in machine learning," in *Advances in Neural Information Processing Systems*, pp. 4148–4158, 2017.

71 K. Pearson, "Notes on regression and inheritance in the case of two parents," in *Proceedings of the Royal Society of London, 58*, pp. 240–242, 1895.

72 S. J. Devlin, R. Gnanadesikan, and J. R. Kettenring, "Robust estimation and outlier detection with correlation coefficients," *Biometrika*, vol. 62, no. 3, pp. 531–545, 1975.

73 W. editors, "Pearson correlation coefficient." https://en.wikipedia.org/wiki/Pearson_correlation_coefficient, 2003.

74 G. Orwell, "Politics and the english language." https://www.orwell.ru/library/essays/politics/english/e_polit, 1946.

75 T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010.

76 X. Zhang and M. Lapata, "Sentence simplification with deep reinforcement learning," *arXiv preprint arXiv:1703.10931*, 2017.

77 Z. Zhu, D. Bernhard, and I. Gurevych, "A monolingual tree-based translation model for sentence simplification," in *Proceedings of the 23rd international conference on computational linguistics*, pp. 1353–1361, Association for Computational Linguistics, 2010.

78 P. Phandi, K. M. A. Chai, and H. T. Ng, "Flexible domain adaptation for automated essay scoring using correlated linear regression," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 431–439, 2015.